NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ANALYSIS OF MECHANISMS FOR TCBE CONTROL OF OBJECT REUSE IN CLIENTS

by

Cihan Agacayak

March 2000

Thesis Advisor:	Cynthia E. Irvine
Second Reader:	William A. Arbaugh

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2000	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Analysis of Mechanisms for TCBE Control of Object Reuse in Clients			**5. FUNDING NUMBERS**
6. AUTHOR(S) Cihan Agacayak.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

This study contributes to the realization of a high assurance Multilevel Secure Local Area Network. The system consists of a Trusted Computing Base (TCB) that acts as a server base. Clients are COTS workstations and software, augmented with a hardware-based TCB Extension (TCBE). This work concentrates on object reuse control on the client, which is one of the security services to be provided by the TCBE. Object reuse mechanisms are designed to assure that sensitive information does not persist across sessions or session level changes. We analyzed 29 chips on the PC motherboard. We proposed and evaluated possible solutions for object reuse control of four storage areas: main memory, AGP memory, cache and Real Time Clock (RTC) memory. The feasibility of one proposed solution was demonstrated. We found that main memory can be cleared by slowing its refresh rate. It was determined that AGP memory cannot be read out by devices on the PCI and ISA bus. The Intel INVD command can be used to clear cache. RTC memory can be accessed and its integrity checked by TCBE software. This study establishes a foundation for object reuse control efforts targeting COTS PC products manufactured by various vendors.

14. SUBJECT TERMS Multilevel Secure Local Area Network (MLS-LAN), Trusted Computing Base (TCB), TCB Extension (TCBE), object reuse, secure systems, object, subject, computers, networking, information security			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFI- CATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std.
239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ANALYSIS OF MECHANISMS FOR TCBE CONTROL OF OBJECT REUSE IN CLIENTS

Cihan Agacayak
Lieutenant Junior Grade, Turkish Navy
B.S.E.E., Turkish Naval Academy, Tuzla Istanbul, 1994

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2000

Author: [illegible]
Cihan Agacayak

Approved by: Cynthia E. Irvine
Cynthia E. Irvine, Thesis Advisor

[illegible]
William A. Arbaugh, Second Reader

[illegible]
Jeffrey Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This study contributes to the realization of a high assurance Multilevel Secure Local Area Network. The system consists of a Trusted Computing Base (TCB) that acts as a server base. Clients are COTS workstations and software, augmented with a hardware-based TCB Extension (TCBE). This work concentrates on object reuse control on the client, which is one of the security services to be provided by the TCBE.

Object reuse mechanisms are designed to assure that sensitive information does not persist across sessions or session level changes. We analyzed 29 chips on the PC motherboard. We proposed and evaluated possible solutions for object reuse control of four storage areas: main memory, AGP memory, cache and Real Time Clock (RTC) memory. The feasibility of one proposed solution was demonstrated.

We found that main memory can be cleared by slowing its refresh rate. It was determined that AGP memory cannot be read out by devices on the PCI and ISA bus. The Intel INVD command can be used to clear cache. RTC memory can be accessed and its integrity checked by TCBE software.

This study establishes a foundation for object reuse control efforts targeting COTS PC products manufactured by various vendors.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENT

I am very grateful for and impressed with the quality of input and support I have received during my efforts to conduct this research. First and foremost, the knowledge and efforts of Dr. Cynthia E. Irvine, my thesis advisor, made the success in this research possible. It is Dr. Irvine's and James P. Anderson's concept of a MLS LAN that initially sparked my interest, and their encouragement kept my motivation high throughout the entire process. I appreciate the guidance, support and enthusiasm provided by James P. Anderson. Many others played integral roles in facilitating this work and deserve special recognition. Paul Clark provided expertise in maintaining the Computer Security Lab LAN. Anastacia Cruz-Tokar provided expertise in graphical tools in the process of creating the figures and tables. I would especially like to thank Dave Shifflett for the brainstorms we had and Tara Lutman for her efforts to make my thesis more readable. Finally, I would like to thank to my parents, Kemal and Aysel Agacayak, who have made numerous sacrifices to bring me where I am now. I would like to devote this thesis to them.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The main purpose of this study is to contribute to the realization of a multilevel secure local area network (MLS-LAN). The system consists of a high assurance Trusted Computing Base (TCB) that acts as a server. The TCB will provide confidentiality by enforcing security policy against unauthorized disclosure of sensitive information while insuring integrity against unauthorized modification and maintaining information reliability. The TCB will allow controlled access to information at multiple security levels and it will support COTS office productivity software at workstations (clients). Clients consist of COTS workstations and software, augmented with a Trusted Computing Base Extension (TCBE). The TCBE will support the MLS-LAN at the client by supplying security services including:

- A secure attention key to invoke trusted path from client to server.
- Control of the PC hardware to provide for object reuse.
- Encryption services for protected communication channels.
- Control of the PC boot process to ensure system integrity.

The TCBE will provide the mechanisms to support the requirements of commercial operating systems and applications while ensuring the enforcement of the network security policy.

This study concentrates on object reuse control on the client, which is one of the security services to be provided by the TCBE.

Why must object reuse be addressed in secure systems? In a secure system, resources are shared under the control of the system. If object reuse is not implemented appropriately then unauthorized access to information may occur. Storage objects could become an information transfer channel between disjoint users as the system reassigns the objects to users. For example, when an object is deleted, the operating system might simply delete the pointer to that object but the data would still exist in the object. For example, if we delete a text file residing in the memory, we actually only delete a pointer which carries the starting address of that file in the memory but the whole text data still exist in the memory. If object access is not controlled appropriately, objects can be accessed by system calls through the use of memory scanners or debugging tools. Once data are accessed, they can be copied onto media accessible to unauthorized individuals.

As resources of the automated information system, objects (such as main memory, cache etc.) store the system's information. The subjects (users) are those who attempt to access system objects. Object reuse mechanisms are designed to assure that the authorized user (subject) of the system doesn't obtain residual information from system resources (objects).

Object reuse is defined as the reassignment of an object to a storage medium (e.g. page frame, disk sector, and magnetic tape) that had contained one or more objects. [Ref. 52]

The Trusted Computer System Evaluation Criteria (TCSEC) [Ref 52] object reuse requirement applies only to storage objects accessible by untrusted users of an automated information system. The Common Criteria [Ref 51] addresses the need to ensure that deleted information is no longer accessible and that newly-created objects do not contain information from previously used objects within the target of evaluation which is, in our case, the PC.

Object reuse becomes very important in client systems since they have multiple users and, without object reuse, the current user may be able to extract some of the previous user's data from the client machine. Even with a user the problem exists. If the user wants to change his or her session level to a less sensitive level, malicious software might transfer sensitive information to this lower sensitivity session level, thereby causing a security breach. This may be done simply by storing sensitive information into one of the objects in the PC, which is preserved across session changes. In this case malicious software might carry out all information transfer undetected by the innocent user.

To be securely reassigned, no residual data can be available to the new subject through standard mechanisms. So, the main goal of object reuse control is to ensure that the allocation and the reassignment of system resources (objects), such as storage to users, be accomplished in such a way as to prevent the disclosure of sensitive information. For example, common physical objects on clients are:

- CPU Registers
- Floating Point Coprocessor Registers
- Peripheral Coprocessor Registers
- Cache Memory

- Physical Memory
- Disk Blocks / Sectors
- Magnetic Tape Blocks / Sectors
- Floppy Disks
- Printer / Scanner Buffers

The common objects implemented by operating systems are:

- Virtual Memory
- Files
- Directories
- Virtual Tape Drives

We are going to be dealing with network security policy not with the security policy of operating systems. Hence we will not investigate the common objects created by the operating systems.

This study intends to examine physical objects in the client PC and to provide feasible solutions to the object reuse control problem by ensuring that there will be no residual secrets in the objects accessible by untrusted entities. This solution will also support start of each new session in a consistent initial state. Starting with a consistent initial state will prevent the client system from possible infection by malicious software (i.e. viruses or Trojan horses) installed in the previous session.

In many organizations, such as the Department of Defense, redundant desktop computer systems are needed since multilevel security levels cannot be utilized in a single workstation environment. The greater the number of security levels, the greater the number of workstations that are required. Employing more workstations to accommodate all security levels increases the cost for the organization.

This study will also contribute to the reduction of the extra cost spent by the Department of Defense (DoD) for redundant desktop computer systems require a separate client machine for the necessary number of sensitivity levels.

Finally, this will help establish a baseline for future miniaturization efforts. The TCBE can be mapped onto a small chip using Very Large Scale Integrated Circuit tools such as those designed by Cadence [Ref. 54]. Thereby enabling us to design handheld computing and communication devices that work on a wireless MLS network. This may provide a significant improvement for the tactical command communications used in operations.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ANALYSIS OF STORAGE AREAS ON MOTHERBOARD

One of the design goals of the TCBE is controlling the object reuse in the PC environment. To accomplish this design requirement, it is important to have an understanding of the storage areas in the PC and the way they are controlled. To do this, the main components of a particular PC supplied for this project were mapped out. This PC will be the host into which our prototype TCBE card will be installed.

The prototype PC has the main components listed below. Each will be analyzed in more detail.

- Motherboard
- Graphics card
- SCSI controller
- Sound card
- Power supply
- Keyboard
- Mouse
- Monitor

A. MOTHERBOARD

The motherboard is the most complex and the main part of the PC. It can be viewed as the heart of the PC. It carries all the components listed above and connects them to each other. These components are connected to each other with lines called "bus lines".

The particular motherboard used in this analysis is a SOYO SY-6BE+ type motherboard. It is a 100 MHz FSB Pentium® II Processor Based ATX Main board with an AGP Port. This motherboard can support 66, 75, 83, 100, 103, 112,124 and 133 MHz system CPU clock speeds. [Ref. 44] [Ref. 45]

It supports the following processors

- 100 MHz FSB Pentium III 450/550 MHz
- 100 MHz FSB Pentium II 350/400/450 MHz
- 66 MHz FSB Pentium II 233/266/300/333 MHz
- 66 MHz FSB Celeron™ 300A~466 MHz
- 66 MHz FSB Celeron™ 266/300 MHz

The motherboard is a high-performance ATX architecture and has software power off control, power-on by keyboard, power on by alarm and modem ring on. It has four 32-bit bus mastering PCI slots, which are PCI version 2.1 compliant. It has three 16-bit ISA slots. One of the ISA slots is a PCI/ISA shared slot. It has one AGP slot, which is version 1.0 compliant.

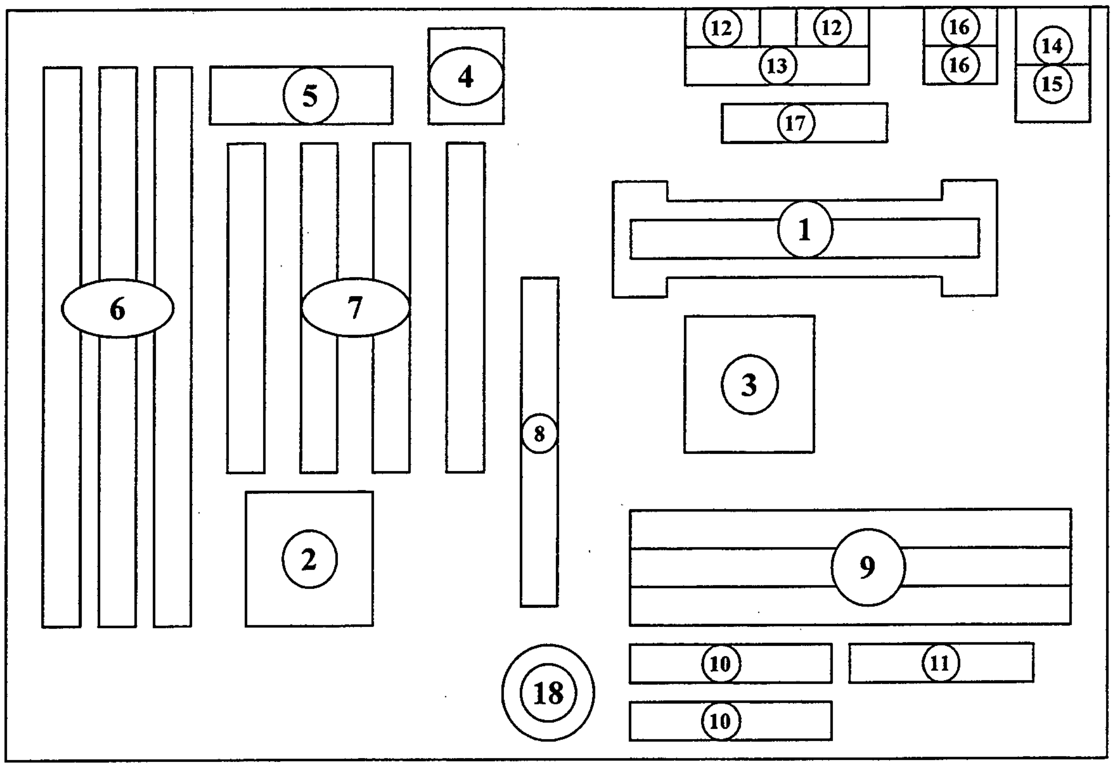

Figure 2.1 motherboard layout from Ref. [44]

Figure 2.1 shows the mapping of the devices residing on the SOYO SY-6BE+ motherboard. The device list associated with this figure is given below.

1. Slot for Pentium II CPU
2. 82371EB Chipset
3. 82443BX Chipset
4. Ultra I/O Chip
5. PnP FLASH BIOS
6. ISA Slots
7. PCI Slots
8. AGP Port
9. DIMM Memory Bank

10. IDE1/IDE2 Connector

11. Floppy Connector

12. COM1/COM2 Connector

13. Parallel Port Connector

14. PS/2 Keyboard Connector

15. PS/2 Mouse Connector

16. USB ½ Connector

17. ATX Power Connector

18. CMOS Battery (Lithium battery, 3V)

Figure 2.2 shows the default jumper and pin settings for the motherboard.

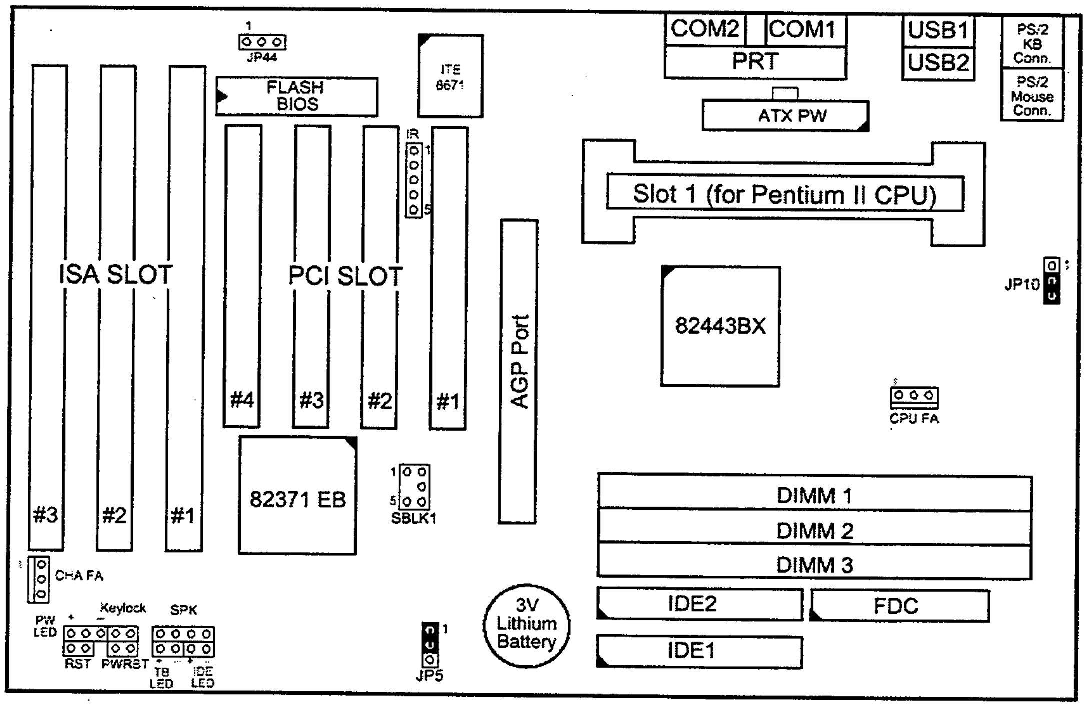

Figure 2.2 motherboard default jumper and pin settings from Ref. [44]

The units on the motherboard are listed below with their serial numbers.

1. CPU Pentium- II 400MHz. (80523PY400512PE SL2U6 99110181-0414 MALAY)

This chip is the brain of the PC. It has eight physical (architectural) registers and cache. The cache structure consists of two level one and one level-two caches. These storage structures are going to be investigated in detail in following chapters. [Ref. 37], [Ref. 17], [Ref. 18]

2. Award BIOS

The BIOS, Basic Input Output System, is the ROM portion of the PC as seen in Figure 2.1. BIOS insulates the system and application software from the hardware by providing primitive I/O services and by programming the hardware's interrupt handling. The BIOS is a read only memory. It doesn't require object reuse control. [Ref. 1]

3. Memory 128 MB SDRAM (Two 64 MB Cards. Each Card Has 8X8 MB Chips. Chip# = 48LC8M8A2)

These two memory chips provide the main system memory for the PC. They are located on the DIMM memory banks shown in Figure 2.1. They will be one of our main object reuse concerns. [Ref. 23]

4. ITE I/O Chips

These chips provide I/O ability to the motherboard. Their location on the motherboard is shown as Ultra I/O chip in Figure 2.1. Detailed information about each individual chip will be given below. [Ref. 35], [Ref. 36]

a. IT8687R/ 9838-EYO/ ZC8U47

This chip is an I/O buffer chip (see Figure 2.3). It can support two RS-232 serial ports and has very low power consumption (150MW). It contains six line drivers and ten line receivers as shown in the block diagram in Figure 2.3. It supports one 24/28 MHz crystal oscillator clock generator.

This low power consumption chip is designed to serve as an interface between data terminal equipment and data communication equipment in conformance with the Electronic Industries Association standard RS-232 specifications. This chip doesn't contain any storage areas.

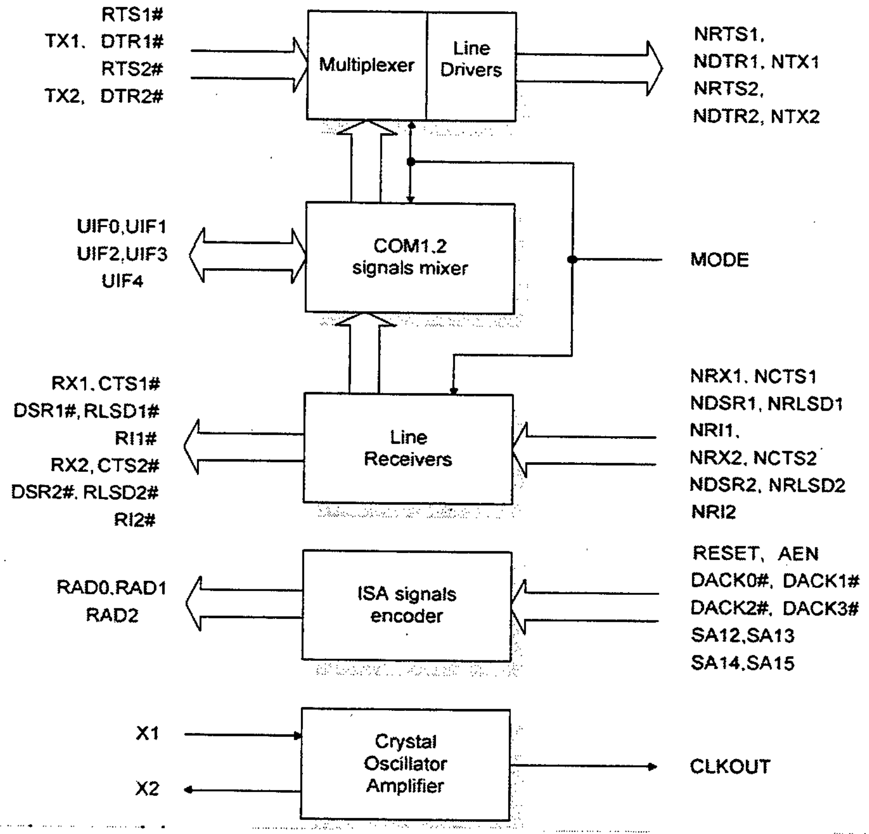

Figure 2.3 IT8687R I/O buffer chip block diagram from Ref. [35]

b. IT8671F-A/ 9837-DYS/ E7X360 I/O buffer chip

The IT8671F Giga I/O is a user-friendly, low cost peripheral controller. With this chip no non-volatile memory is needed to store resource data for Plug and Play system applications.

This chip consists of five logical devices (see Figure 2.4). The first is a high-performance 2.88MB floppy disk controller with a digital data separator, supports two 360K/ 720K /1.2M /1.44M /2.88M floppy disk drives.

The second is a multi-mode high-performance parallel port (see Figure 2.4) that features the bi-directional Standard Parallel Port (SPP), the Enhanced Parallel Port (EPP, v1.7 and v1.9 are supported), and the IEEE 1284 compliant Extended Capabilities Port (ECP).

The third and the fourth are two 16C550 standard compatible enhanced UARTs (see Figure 2.4) that perform asynchronous communication with enhanced wireless IrDA1 (HPSIR), MIR, FIR or ASKIR protocols.

Finally, there is one 8042 compatible Keyboard controller with 2K programmable ROM for customer specification. This chip has a 2K programmable ROM and 256 bytes of data RAM in the keyboard controller module.

These five logical devices can be individually enabled or disabled via software configuration registers.

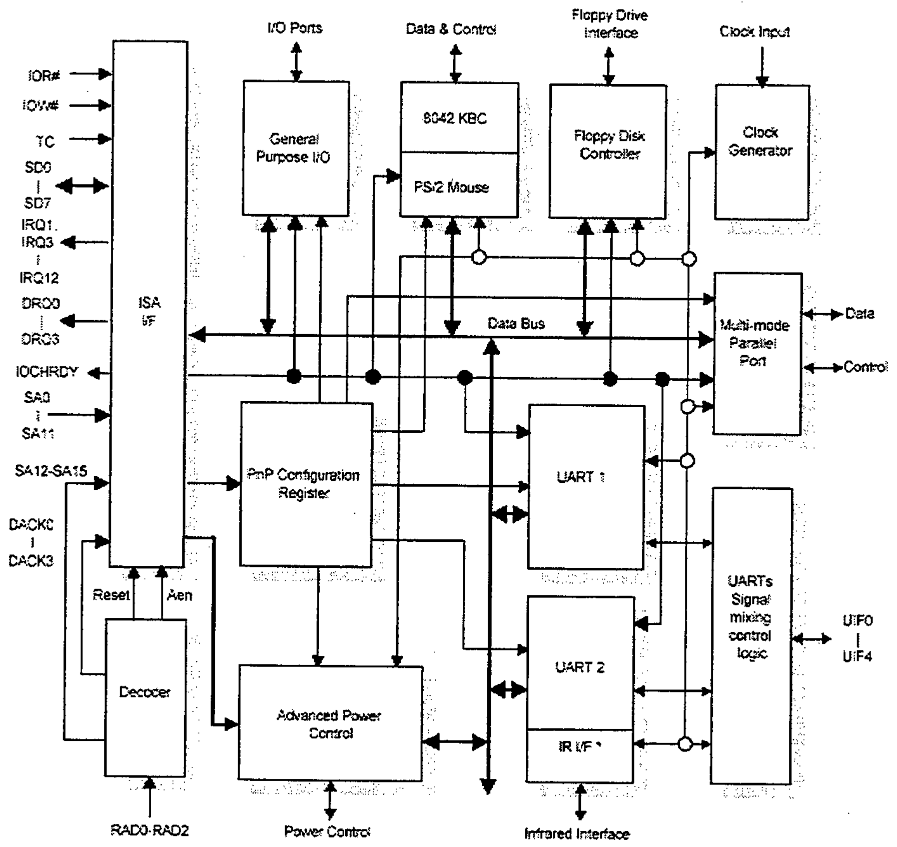

Figure 2.4 IT8671F Giga I/O block diagram from Ref. [36]

It also has configuration registers in the logical devices but they are returned to their default values after a reset signal reception. In this device our main concern about object reuse will be the 256 bytes of data RAM in the keyboard controller module.

5. 82443BX PCI/Host AGP Controller

This chip is the part of the 440BX AGP set, as shown in Figure 2.1. The detailed block diagram for this chipset is given in Figure 2.5. It has been designed to interface between the Pentium II processor's system bus at 100 or 66 MHz. This chip is a Host-to-

PCI Bridge interface but also it has been optimized with a 100/66 MHz SDRAM memory controller and data path. [Ref.4]

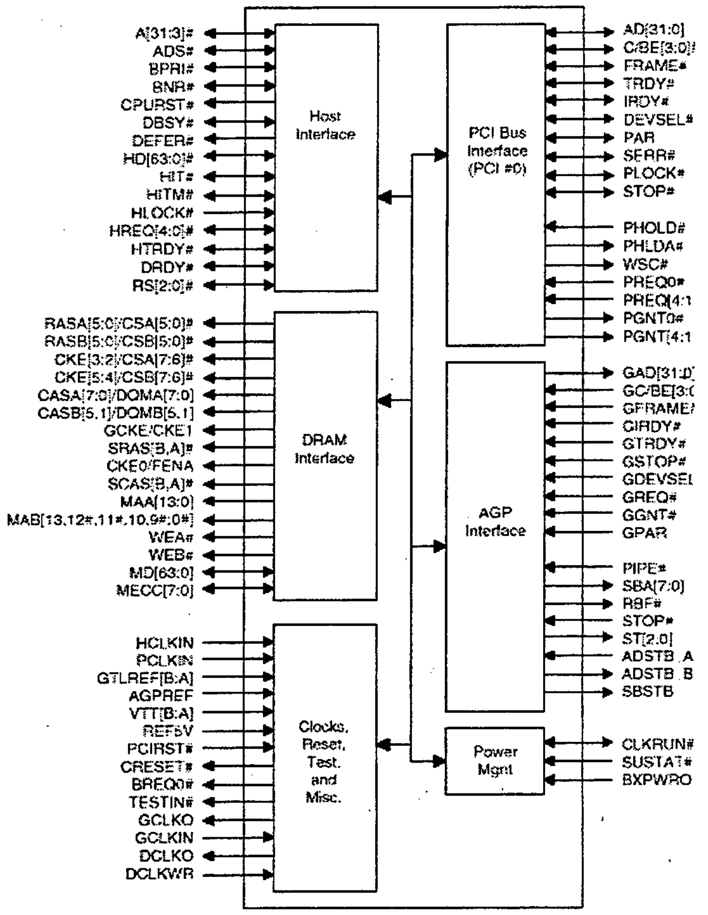

Figure 2.5 82443BX block diagram from Ref. [4]

This chip also has the Accelerated Graphics Port (AGP) interface functionality. AGP is a high performance, component-level interconnect targeted at 3D graphics applications and is based on a set of performance enhancements to PCI. This chip doesn't have any storage areas other than the configuration registers.

6. 82371AB PCI/ISA/IDE Accelerator (PIIX4)

This chip is the part of the 440BX AGP set. [Ref. 5] describes the 82371AB PCI/ISA/IDE accelerator (PIIX4) as a multifunction PCI device (see Figure 2.6). It implements a PCI-to-ISA bridge function, a Universal Serial Bus host/hub function, and an Enhanced Power Management function.

As a PCI-to-ISA bridge, the PIIX4 integrates many common I/O functions found in ISA-based PC systems, two 82C37 DMA Controllers, two 82C59 Interrupt Controllers, an 82C54 Timer/Counter, and a real time clock. Chip select decoding is provided for the BIOS, real time clock, keyboard Controller, second external microcontroller, as well as two Programmable Chip Selects. The PIIX4 provides full Plug and Play compatibility.

The PIIX4 supports two IDE connectors for up to four IDE devices providing an interface for IDE hard disks and CD ROMs. Up to four IDE devices can be supported in Bus Master mode. The PIIX4 contains support for "Ultra DMA/33" synchronous DMA compatible devices.

The PIIX4 contains a Universal Serial Bus (USB) Host Controller that is Universal Host Controller Interface (UHCI) compatible. The Host Controller's root hub has two programmable USB ports.

The PIIX4 supports Enhanced Power Management, including full Clock Control, Device Management for up to 14 devices, and Suspend and Resume logic with Power-on Suspend, Suspend to RAM or Suspend to Disk. It fully supports Operating System Directed Power Management via the Advanced Configuration and Power Interface (ACPI) specification.

This chip has the real time clock memory, which is a considerable amount of storage area, 256-bytes in size.

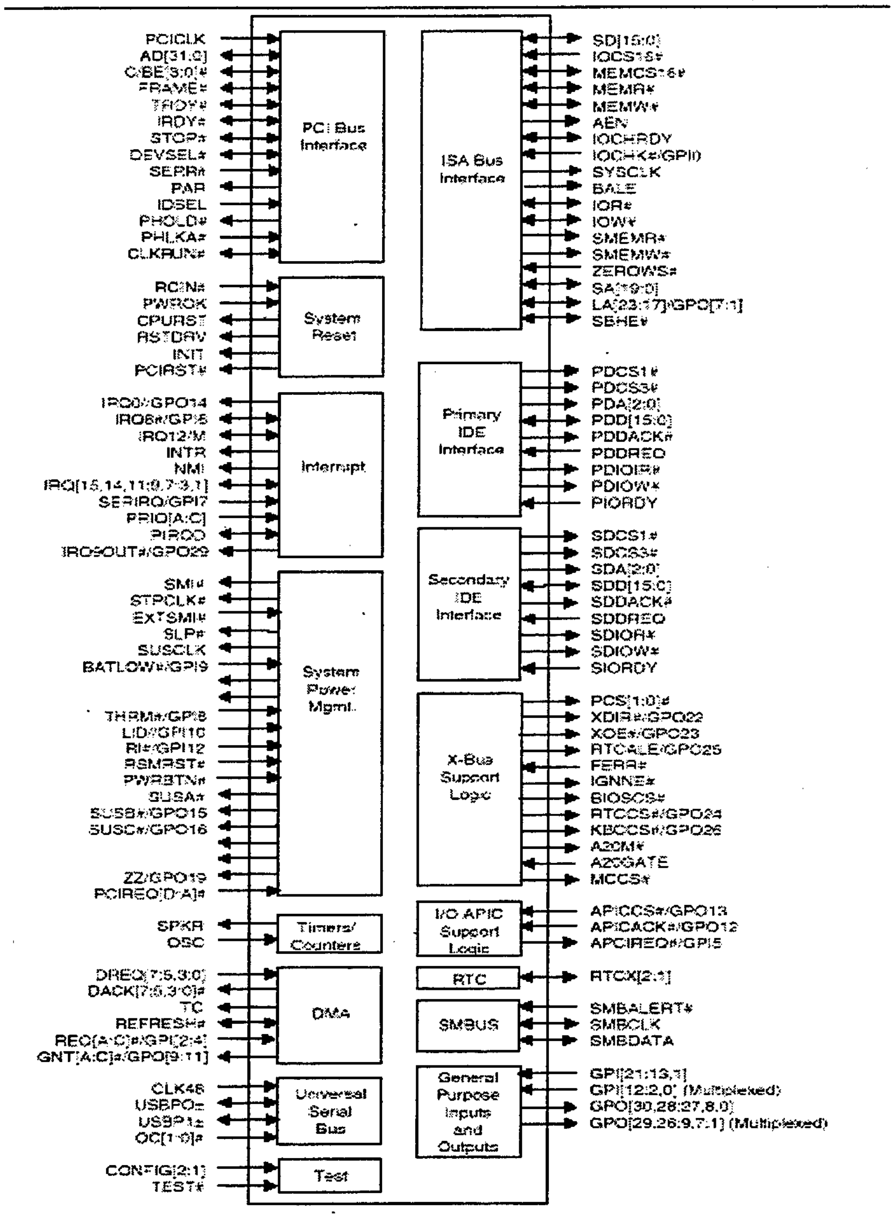

Figure 2.6 82371AB block diagram from Ref. [5]

7. Other Existing Chips

The chips that are listed in this section are physically very small sized and unevenly distributed on the motherboard. To keep the general picture simple we didn't show the location of these chips in Figure 2.1 and in Figure 2.2.

a. *W83781D / 836AC Winbond Hardware Monitoring IC*

This chip is a hardware status monitoring IC, shown in Figure 2.7, for personal computers, server computers or microprocessor-based systems. It monitors the critical values for the system such as the power supply voltage, temperature and fan tachometer readings. It also has a case open alarm. This chip can be controlled by Intel's LAN Desk Client Management or Winbond's application software. It has 33 registers and 20 of them are read/write [Ref. 34]. There is a total storage area of 20 bytes in this chip. In this study we deferred analysis of this chip

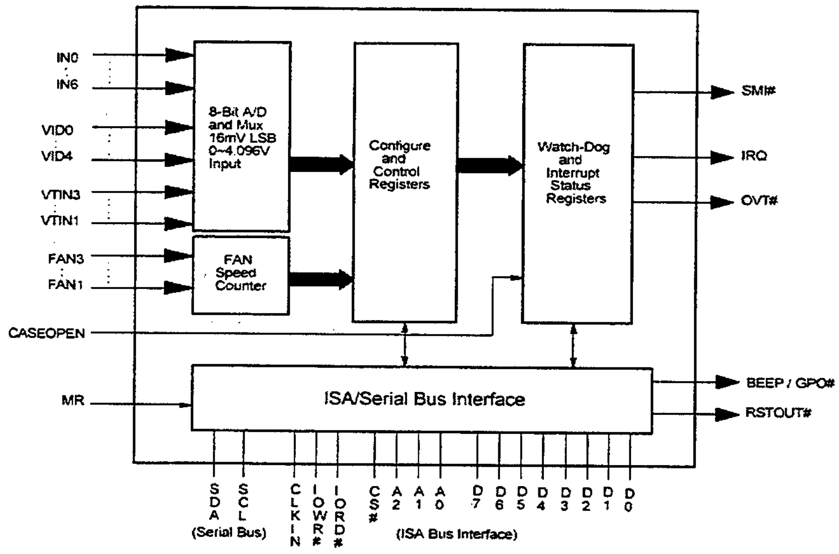

Figure 2.7 W83781D / 836AC block diagram from Ref. [34]

b. *74F174D / 6CK6709 Edge Triggered D-Type Flip-Flop*

This chip is a high-speed edge triggered D-type flip-flop, shown in Figure 2.8. It is used primarily as a 6-bit edge-triggered storage register. The information on the D inputs is transferred to storage during the LOW-to-HIGH clock transition. The device has a master reset to simultaneously clear all flip-flops. This chip doesn't have a storage area. [Ref. 24]

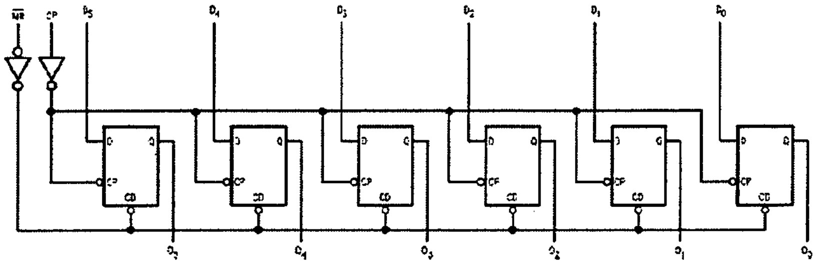

Figure 2.8 74F174D / 6CK6709 logic diagram from Ref. [24]

c. *84AY33K / LV244A Buffer/Line Driver*

This component is a low voltage Si-gate CMOS device, shown in Figure 2.9. It is an octal non-inverting buffer/line driver with 3-State outputs. This chip doesn't have any storage structures on it. [Ref. 26]

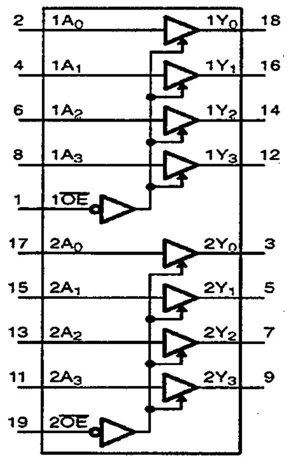

Figure 2.9 84AY33K / LV244A logic diagram from Ref. [26]

d. W124G/B014/1833PH Motherboard Frequency Generator

This chip is a 100MHz-spread spectrum motherboard frequency generator. This chip doesn't have any storage area.

e. DM7407M Hexadecimal Buffer/Driver

This chip is a hexadecimal buffer/driver with high voltage open-collector outputs, shown in Figure 2.10. It contains six independent gates, each of which performs a buffer function. This chip doesn't contain any storage areas. [Ref. 27]

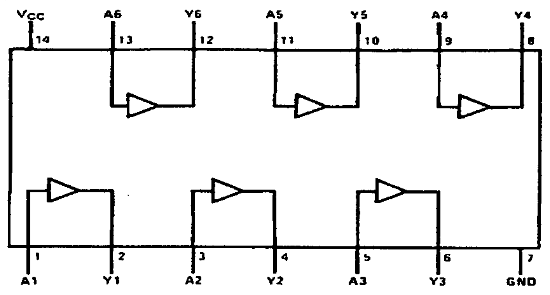

Figure 2.10 DM7407M block diagram from Ref. [27]

f. LM2635M Synchronous Buck Regulator Controller

This chip is a 5-bit programmable synchronous buck regulator controller, shown in Figure 2.11. It is specifically designed for use in synchronous DC/DC buck converters for the Pentium II or Deschutes microprocessor. It provides power good signal, over-voltage protection and output enable features as required by Intel VRM specifications. This chip doesn't have any storage areas. [Ref. 22]

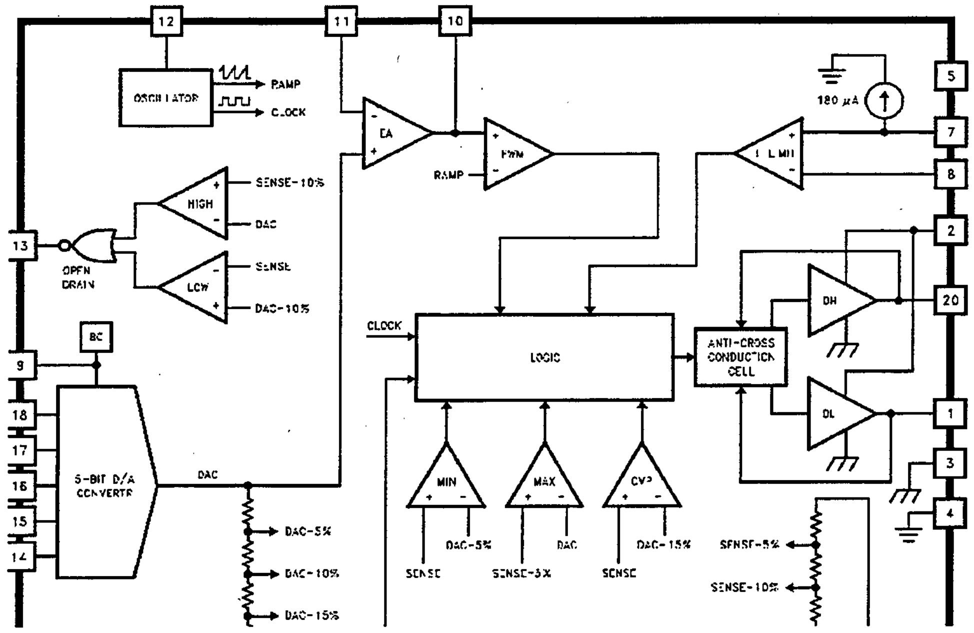

Figure 2.11 LM2635M block diagram from Ref. [22]

g. *DM74ALS05A Hexadecimal Inverter*

This chip is a-hexadecimal inverter with open collector outputs, shown in Figure 2.12. This device contains six independent gates, each of which performs the logic INVERT function. The open-collector outputs require external pull-up resistors for proper logical operation. This chip doesn't have any storage areas. [Ref. 29]

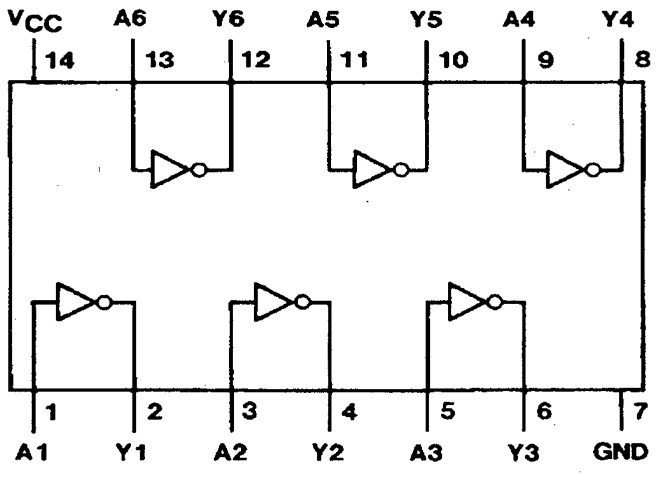

Figure 2.12 DM74ALS05A block diagram from Ref. [29]

h. DM74ALS08A Quad 2-Input AND Gate

This chip is a quad 2-input AND gate, shown in Figure 2.13. It contains four independent gates, each of which performs the logic AND function. This unit doesn't have any storage capability. [Ref. 30]

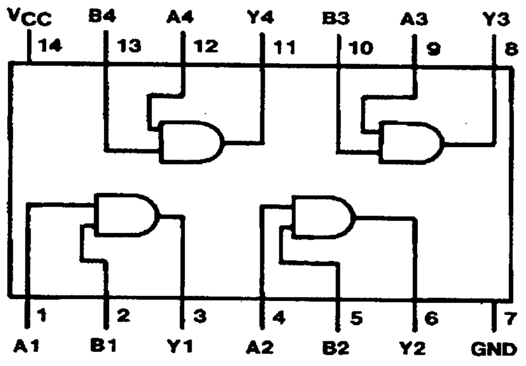

Figure 2.13 DM74ALS08A block diagram from Ref. [30]

i. 74HCT14 / 90K806 Hexadecimal Schmitt-Trigger Inverter

This chip is a hexadecimal Schmitt-Trigger Inverter with LSTTL compatible inputs, shown in Figure 2.14. This device can be used as a level converter for interfacing TTL or NMOS outputs to high-speed CMOS inputs. The 74HCT14 is useful to square up slow input rise and fall times. Due to the hysteresis voltage of the Schmitt trigger, the HCT 14A finds applications in noisy environments. This chip doesn't have any storage capability. [Ref. 25][Ref. 32]

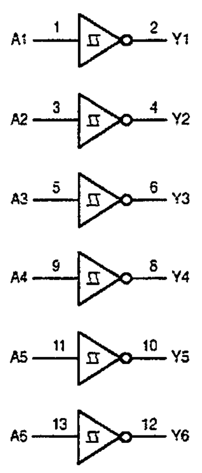

Figure 2.14 74HCT14 / 90K806 block diagram from Ref. [25]

j. NE555 Oscillator

This chip is a highly stable device for generating accurate time delays or oscillations, shown in Figure 2.15. Additional terminals are also provided for triggering or resetting. In the time delay mode, one external resistor and capacitor precisely controls the time. This chip can make timing oscillations from microseconds through hours. This chip doesn't have any storage area. [Ref. 31]

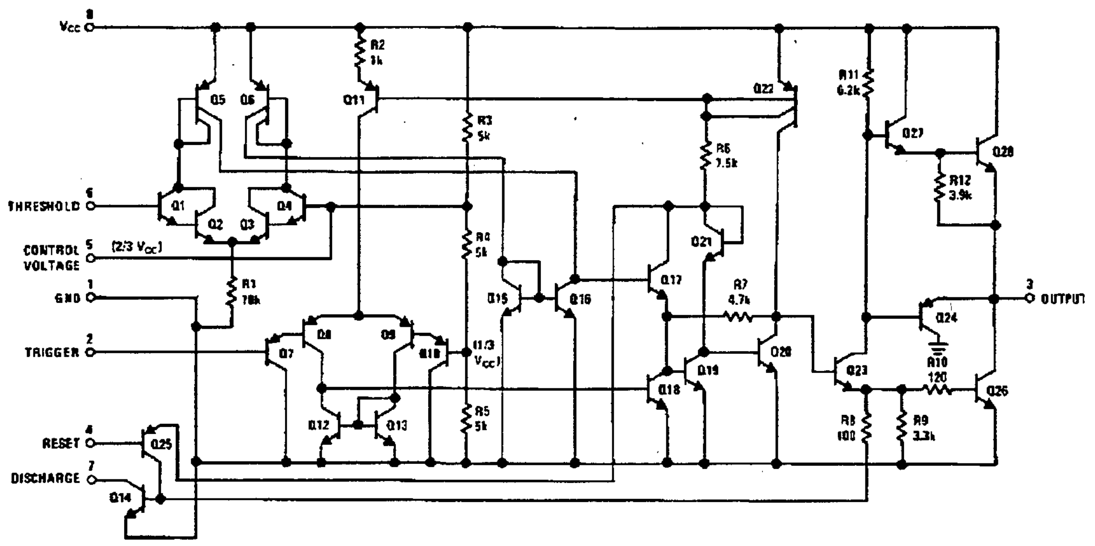

Figure 2.15 NE555 block diagram from Ref. [31]

k. W40S11-23G/B871 1832PD Clock Buffer/Driver

This chip on the PC is a low voltage, thirteen-output clock buffer/driver, shown in Figure 2.16. Since the output buffer impedance is approximately 15Ω, this device is ideal for driving SDRAM DIMMs and it doesn't have any storage capability. [Ref. 33]

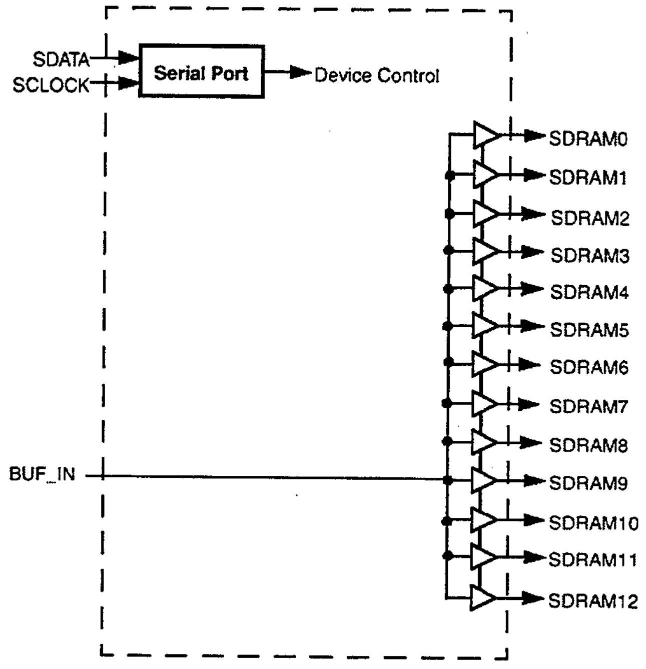

Figure 2.16 W40S11-23G/B871 1832PD block diagram from Ref. [33]

B. GRAPHICS CARD

1. HOLTEK HT27C5120-70 / 9852K0532-2

This chip family is a low power, 512 Kbit, +5V electrically one–time programmable EPROM, shown in Figure 2.17. It is organized into 64K words with 8 bits per word. It features a fast single address location programming, typically 75μs per byte (write access). Any byte can be accessed (read) in less than 70ns/90ns with respect to the specifications. This eliminates the need for WAIT states in high-performance microprocessor systems [Ref. 19]. This chip is a one time writable chip, so the storage area in this chip is read only.

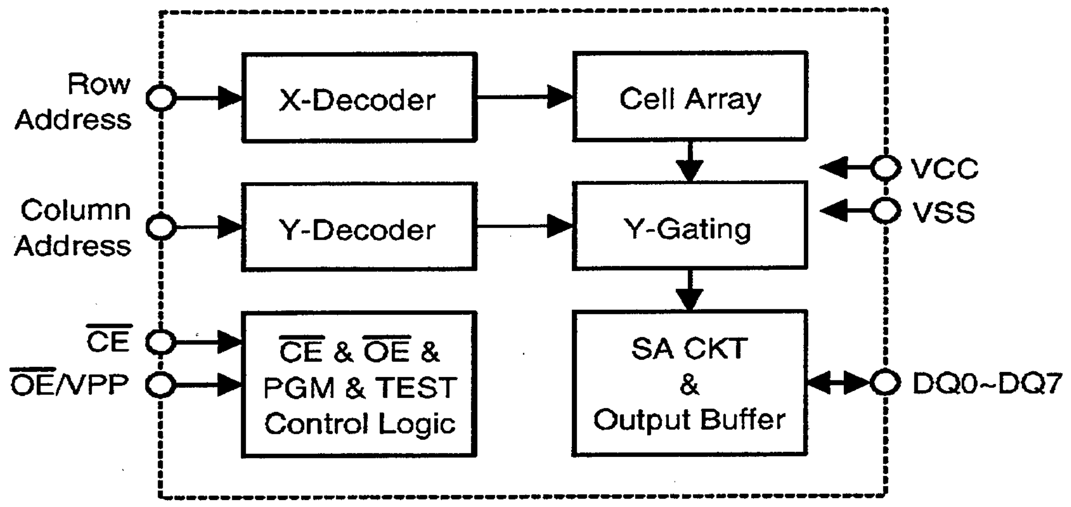

Figure 2.17 HOLTEK HT27C5120-70 block diagram from Ref. [19]

2. 5002LC / NPC

This chip is a monolithic, wideband, high slew rate (fast response), and high output current buffer amplifier (shown in Figure 2.18). It offers a slew rate within 110MHz of bandwidth. It is known as a very reliable device and it increases the overall circuit performance. This chip doesn't have any storage capability. [Ref. 21]

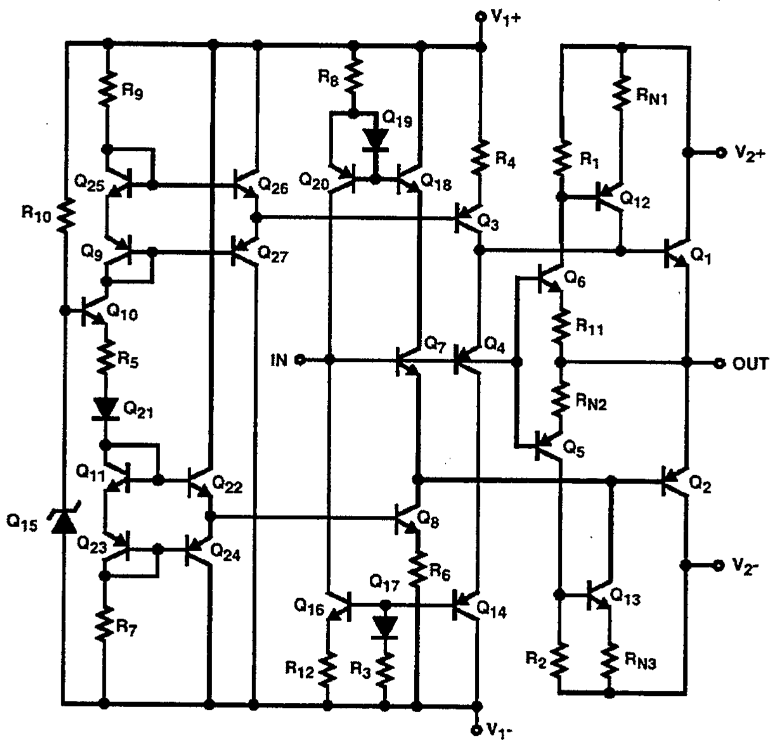

Figure 2.18 5002LC / NPC block diagram Ref. [21]

3. 1X16LKTW-SS (Four of Them)

This device is a memory module used as the local memory for the AGP card. The usual size of this memory changes from 2 to 8 MB. This memory is a SRAM type memory and it is a non-volatile storage area.

4. Unidentified Chip

This device couldn't be identified due to the heat sink attached on the chip and no documentation is currently available. It is suspected that it could be the chip HT82V167-100QFP [Ref. 20]. The HT82V167/HT82V168 VCD-plus A/V decoder is an enhanced version of VCD Audio/Video decoder IC.

C. SCSI CONTROLLER

1. L Infinity LX5115CD/ 9908C

The L Infinity LX5115CD/ 9908C chip is an ultra 9-line SCSI terminator (shown in Figure 2.19). Recognizing the needs for portable and configurable peripherals, the LX5115 has a TTL compatible sleep/disable mode. This architecture can implement 8-bit or 16-bit wide applications. It is approved for use with SCSI 1,2,3 and Ultra SCSI standards. This device doesn't have any storage capability. [Ref. 42]

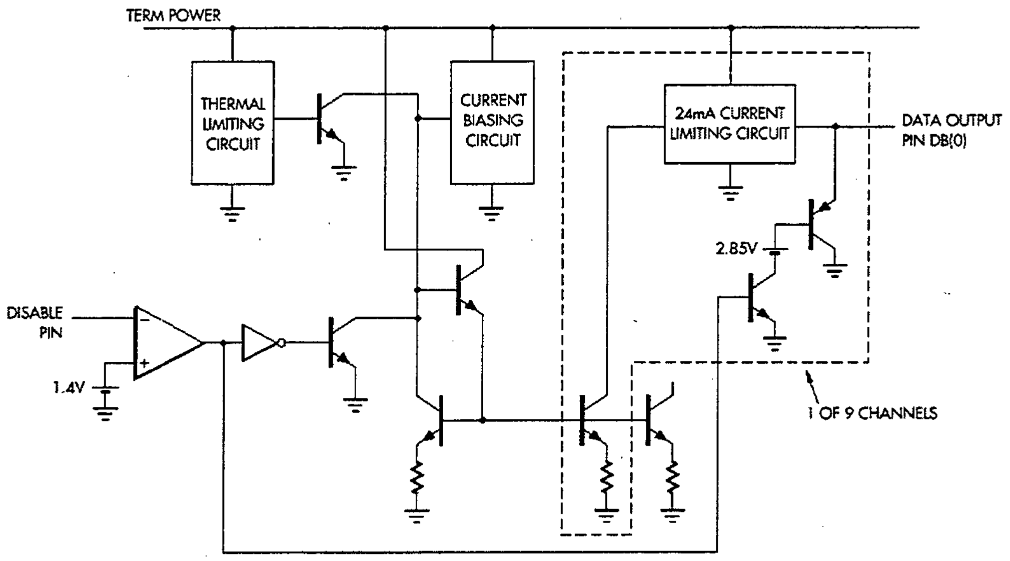

Figure 2.19 L INFINITY LX5115CD/ 9908C block diagram from Ref. [42]

2. CSI 93C46S / 9907G

The CSI 93C46S / 9907G chip is a one Kbit serial EEPROM memory device (shown in Figure 2.20). It is configured as either registers of 16 bits or 8 bits. Each register can be written or read serially. This device is designed to endure 1,000,000 program/erase cycles and have data retention of 100 years [Ref. 41]. This chip has one

Kbit capacity and this storage area is hardware write protected. Because of this hardware write protection we don't have to worry about the object reuse control of this storage area.

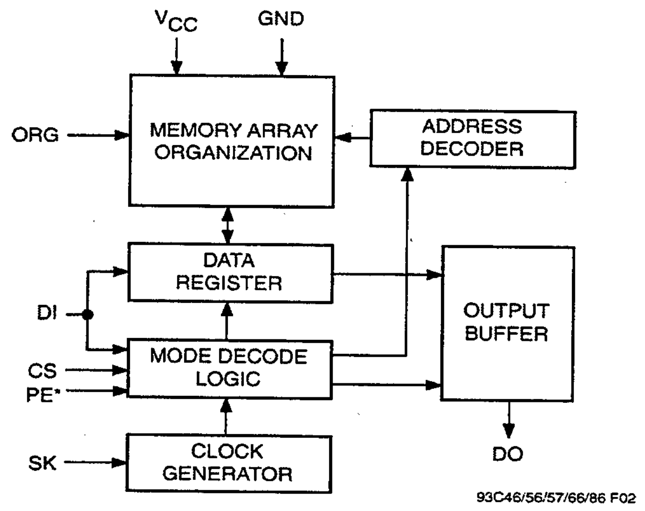

Figure 2.20 CSI 93C46S / 9907G block diagram from Ref. [41]

3. Adaptec AIC-7856T/ BQEB910 / 745011 / BK 1965.1

This device is a single chip PCI to fast SCSI controller. This chip works as an interface between the PCI protocol and the SCSI protocol. Currently there is no public documentation available which provides enough information about any existing storage area in this chip.

4. BIOS 7B00 / 1701301-00A / V1.34.1 / Year 1998

This chip is the BIOS of the SCSI controller card. Since this chip is a ROM, we don't need to worry about the object reuse control of this storage area.

D. SOUND CARD

1. Avance Logic, Inc. / ALS4000 / 93197T1 913C

This chip is a 16-bit full duplex sound controller chip. Currently there is no public documentation available which provides enough information about any existing storage area in this chip.

2. HA17358

This chip is a low power dual operational amplifier for the sound card. Currently there is no public documentation available which provides enough information about any existing storage area in this chip.

E. POWER SUPPLY

The power supply provides and arranges the voltage and current levels for the PC. It does not have any storage areas directly connected to the client PC.

F. KEYBOARD

UM6868-099649M MA24G5:

This chip is the decoder for the keyboard keystrokes. It doesn't have any storage areas in it.

G. NETWORK ADAPTER CARD

The network card provides connection to an existing network. Currently a network card is not installed on the motherboard of the client PC.

We need to analyze these components in order to find the storage areas, which can be involved in the object reuse process. In this chapter the study concentrated on the storage areas in the PC environment in the form of RAM. There are also other storage areas, which consist of registers, and buffers as we can see from the detailed chip mapping of the prototype PC. In this study we will not cover these minor storage areas for object reuse control. The main storage areas found after this analysis are as given in Table 2.1.

	RAM	SDRAM	EEPROM	ROM	EPROM	SRAM
MOTHER BOARD	GIGA I/O	48LC8M8A2	AWARD BIOS			P-II L1-L2 CACHE
GRAPHICS CARD					HOLTEK HT27C5120-70 / 9852K0532-2	1X16LKTW -SS
SCSI CONTROLLER				1701301-00A BIOS 7800 V1.34.1		
CMOS CHIP	82371 AB					

Table 2.1 Storage Areas in the PC

THIS PAGE INTENTIONALLY LEFT BLANK

III. PC MOTHERBOARD AND CHIPSET OPERATION BASICS

To control object reuse in a PC, we need to understand how the machine works. The secret of the operation of the PC lies in its heart: the chipset. To understand chipset functions we need to know about the communication happening on the motherboard and the central processor unit embedded on it. Each device in the PC communicates with other devices over a protocol. In this chapter we are going to investigate how components in a PC get initialized and how the PCI protocol works.

A. PC INITIALIZATION

We need to understand the start up process of the computer. This information will assist us in defining some of the design aspects of the TCBE.

In this section we will talk about a generic startup of an IBM compatible computer as it is also described in [Ref. 1][Ref. 6][Ref. 7][Ref. 8][Ref. 9][Ref. 10][Ref. 11][Ref. 38][Ref. 39][Ref. 40]. Every startup event differs slightly depending on the vendor and the hardware configuration of the PC.

Pressing the power on button on the PC starts the first spark. When the machine is first powered on the voltage outputs of the power supply are not at the correct levels yet. To prevent any devices from operating until the power has stabilized, the power supply keeps its "power good" output signal at deasserted during this period. The deasserted signal is inverted on the motherboard side and used as an asserted system reset signal. The reset signal is propagated into all devices so that no device can operate until this signal is deasserted. The reset signal remains asserted until the power is stabilized. The power supply asserts the "power good" signal when the power has stabilized. On the motherboard side this causes the reset signal to be deasserted. Besides preventing devices from operating, the reset signal also presets all the devices to a known state so that they always start operating the same way when the reset is removed.

This forces the x86 processor to always come up in real mode with caching, paging and interrupts disabled. When the reset is deasserted the processor fetches its first instruction from ROM memory. The power on start address is 0000FFFFh for the CPU. This address contains an unconditional branch to an address lower in memory. These locations contain the first code lines of the power on self-test (POST) and configuration program. At the beginning the caching is disabled. When caching is disabled the prefetcher in the CPU always does 32 byte code reads.

In a multiprocessor environment only one of the processors is selected to begin fetching and executing the POST when reset is removed from the processors. The same processor also will configure the system board devices and enable them, detect the existence of other processors and perform the boot process to read the operating system into memory and pass the control to it. This processor is called the bootstrap processor (BSP). To define the BSP the processors negotiate amongst themselves before the first instruction is fetched from memory. This negotiation is not performed on the host processor bus. It is performed on a special bus, which is called as the Advanced Programmable Interrupt Controller. After the BSP is defined all other processors are defined as application processors and they remain in the halt state until they get a startup message from the BSP.

When the machine is started the majority of the devices in the machine are disabled. Devices that are power up enabled must be operational when the machine is first started. For example, the keyboard should be responsive at start up. The display must be enabled in text mode so that it can display messages related with the flow of the process and emerging warnings and errors during the startup process. The mass storage controller must be enabled so that the programs can be loaded into memory and executed.

The chipset and the memory controllers are configured using the configuration mechanism, which is also used for the configuration of the PCI devices. This may seem confusing since the memory controller doesn't truly reside on the PCI bus. Each of the chipset members and the memory controller implement the PCI configuration address port and configuration data port. We are going to analyze the PCI configuration mechanism in detail later.

The POST and hardware initialization process is the result execution of the code residing in the BIOS. On our prototype computer we have the AWARD Modular BIOS version 4.51-PG.

For more detailed information about the POST and initialization process the steps taken by the BIOS code for our prototype computer are given in Table 3.1 below. Here again we note that these steps could be in different type or order according to the vendor and hardware configuration.

Code (hex)	Name	Description
Co	Turn Off Chipset	Cache OEM Specific-Cache control
1	Processor Test 1	Processor Status (1FLAGS) Verification. Tests the following processor status flags: carry, zero, sign, and overflow. The BIOS sets each flag, verifies they are set, then turns each flag off and verifies it is off.
2	Processor Test 2	Read/Write/Verify all CPU registers except SS, SP, and BP with data pattern FF and 00.
3	Initialize Chips	Disable NMI, PIE, AIE, UEI, and SQWV. Disable video, parity checking, DMA. Reset math coprocessor. Clear all page registers, CMOS shutdown byte. Initialize timer 0, 1, and 2, including setting EISA timer to a known state. Initialize DMA controllers 0 and 1. Initialize interrupt controllers 0 and 1. Initialize EISA extended registers.
4	Test Memory Refresh Toggle	RAM must be periodically refreshed to keep the memory from decaying. This function ensures that the memory refresh function is working properly.
5	Blank video, Initialize keyboard	Keyboard controller initialization.
6	Reserved	
7	Test CMOS Interface and Battery Status	Verifies CMOS is working correctly, detects bad battery
BE	Chipset Default Initialization	Program chipset registers with power on BIOS defaults.
C1	Memory presence test	OEM Specific-Test to size on-board memory
C5	Early Shadow	OEM Specific-Early Shadow enable for fast boot.
C6	Cache presence test	External cache size detection
8	Setup low memory	Early chip set initialization. Memory presence test. OEM chip set routines. Clear low 64K of memory. Test first 64K memory.
9	Early Cache Initialization	Cyrix CPU initialization. Cache initialization
A	Setup Interrupt Vector Table	Initialize first 120 interrupt vectors with SPURIOUS-INT-HDLR and initialize INT 00h-lFh according to INT_TBL
B	Test CMOS RAM Checksum	Test CMOS RAM Checksum, if bad, or insert key pressed, load defaults.
C	Initialize keyboard	Detect type of keyboard controller (optional) Set NUM-LOCK status.
D	Initialize Video Interface	Detect CPU clock. Read CMOS location 14h to find out type of video in use. Detect and Initialize

		Video Adapter.
E	Test Video Memory	Test video memory, write sign-on message to screen. Setup shadow RAM - Enable shadow according to Setup
F	Test DMA Controller 0	BIOS checksum test. Keyboard detect and initialization
10	Test DMA Controller 1	
11	Test DMA Page Registers	Test DMA Page Registers
12-13	Reserved	
14	Test Timer Counter 2	Test 8254 Timer 0 Counter 2.
15	Test 8259-1 Mask Bits	Verify 8259 Channel 1 masked interrupts by alternately turning off and on the interrupt lines.
16	Test 8259-2 Mask Bits	Verify 8259 Channel 2 masked interrupts by alternately turning off and on the interrupt lines.
17	Test Stuck 8259's Interrupt Bits	Turn off interrupts then verify no interrupt mask register is on.
18	Test 8259 Interrupt Functionality	Force an interrupt and verify the interrupt occurred
19	Test Stuck NMI Bits (Parity/IO Check)	Verify that NMI can be cleared.
1A	Display CPU clock	
1B-IE	Reserved	
1F	Set EISA Mode	If the EISA non-volatile memory checksum is good, execute EISA initialization. If not, execute ISA tests and clear the EISA mode flag. Test EISA Configuration Memory Integrity (checksum and communication interface).
20	Enable Slot 0	Initialize slot 0 (System Board).
21-2F	Enable Slots1-15	Initialize slots 1 through 15.
30	Size Base and Extended Memory	Size base memory from 256K to 640K and extended memory above 1MB.
31	Test Base and Extended Memory	Test base memory from 256K to 640K and extended memory above 1MB using various patterns. NOTE: This test is skipped in EISA mode and can be skipped with ESC key in ISA mode.
32	Test EISA Extended Memory	If the EISA Mode flag is set then test, EISA memory found in slots initialization. NOTE: This test is skipped in ISA mode and can be skipped with ESC key in EISA mode.
33-3B	Reserved	
3C	Setup Enabled	
3D	Initialize & Install	Detect if the mouse is present, initialize the mouse,

	Mouse	install interrupt vectors.
3E	Setup Cache Controller	Initialize cache controller
3F	Reserved	
BF	Chipset Initialization	Program chipset registers with Setup values
40		Display virus protect disable or enable
41	Initialize Floppy Drive and Controller	Initialize floppy disk drive controller and any drives.
42	Initialize Hard Drive and Controller	Initialize hard drive controller and any drives.
43	Detect & Initialize Serial/Parallel Ports	Initialize any serial and parallel ports (including the game port).
44	Reserved	
45	Detect & Initialize Math Coprocessor	Initialize the math coprocessor
46	Reserved	
47	Reserved	
48-4D	Reserved	
4E	Manufacturing POST Loop or Display Messages	Reboot if the Manufacturing POST Loop pin is set. Otherwise display any messages (i.e., any non-fatal errors that were detected during POST) and enter Setup.
4F	Security Check	Ask password security (optional).
50	Write CMOS	Write all CMOS values back to RAM and clear screen.
51	Pre-boot Enable	Enable parity checker. Enable NMI, Enable cache before boot
52	Initialize Option ROMs	Initialize any option ROMs present from C8000h to EFFFFh. NOTE: When the FSCAN option is enabled, ROMs initialize from C8000h toF7FFFh.
53	Initialize Time Value	Initialize time value in 40h: BIOS area.
60	Setup Virus Protect	Setup virus protect according to Setup
61	Set Boot Speed	Set system speed for boot
62	Setup NumLock	Setup NumLock status according to Setup
63	Boot Attempt	Set low stack. Boot via INT 19h.
BO	Spurious	If interrupt occurs in protected mode
B1	Unclaimed NMI	If unmasked NMI occurs, display Press Fl to disable NMI, F2 reboot.
El-EF	Setup Pages	El- Page 1, E2 - Page 2, etc.
FF	Boot	

Table 3-1 AWARD BIOS POST Codes from Ref. [39]

NOTE: EISA POST codes are typically output to port address 300h. ISA POST codes are output to port address 80h.

By looking at this table we can gain an understanding of the startup process of an IBM compatible computer.

The TCBE is planned to be designed as a PCI add on card. To define some of the design aspects of the TCBE we need to understand what PCI means and how it works.

B. WHAT IS PCI

This section presents the PCI standard, which is to be used in the key solutions related to object reuse control efforts. PCI stands for "Peripheral Component Interconnect". Intel Corporation developed the PCI bus specification version 1.0. A consortium of industry partners known as the PCI special interest group (SIG) now manages the specification. The latest revision of the specification is 2.2 and our prototype computer supports this revision. The PCI bus can be populated with adapters requiring fast access to each other and/or system memory. They can be accesses by the processor at the full native bus speed. Also note that all read and write transactions over the PCI bus can be done as burst transactions, increasing the transaction speed.

There are two participants in every PCI burst transfer, the initiator and the target. The initiator can also be called the bus master. The target is the device addressed by the bus master. PCI initiator and target devices are commonly referred to as PCI-compliant agents according to the PCI spec.

Burst transaction means that every data transfer doesn't need to be preceded by address information. In a burst transaction a single address phase can be followed by two or more data phases. The target is given the start address and the type of the transaction. The initiator informs the target whether the coming data is the last one or not. The transaction completes when the final data arrives at the target.

Figure 3.1 below illustrates the relation between the PCI, expansion, processor and memory buses.

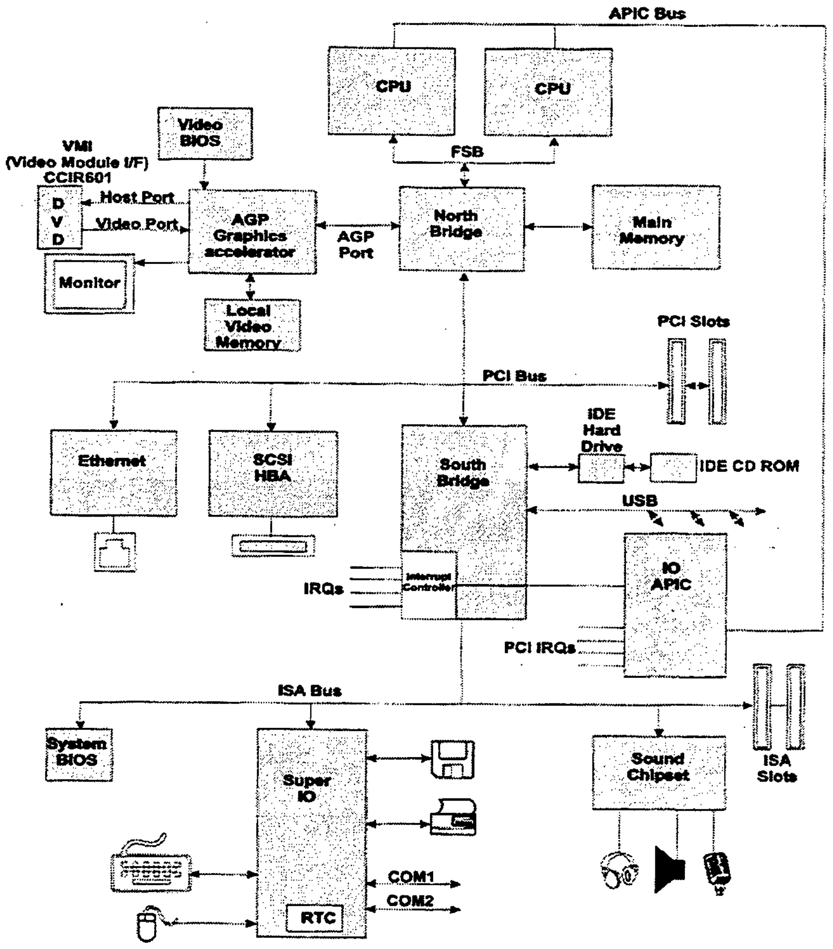

Figure 3.1 PCI system architecture in a PC from Ref. [10]

In Figure 3.1 the North Bridge is the Host/PCI Bridge and it connects the host processor bus to the PCI bus. The South Bridge is the PCI-to-ISA Bridge. South bridge connects the PCI bus to the ISA or EISA bus. The south bridge also utilizes the interrupt controller, IDE controller, USB Host controller and the DMA controller. All actions on the PCI bus are synchronized to the PCI clock signal. The PCI revision 1.0 stated that all PCI devices must support operation speeds from 16 MHz up to 33 MHz. The PCI revision spec 2.1 also defined PCI bus operation at speeds up to 66 MHz. Now we will investigate how the PCI devices are detected.

C. HOW IS THE PCI DEVICE DETECTED BY THE SYSTEM

A PCI device may either be embedded on the PCI bus or installed in a PCI add-in connector. In either case each device is assigned a physical device number based on its physical position on the bus.

A PCI device is detected by attempting to read from its vendor ID register. This is a required 16-bit register. If the target is present a vendor ID other than FFFFh is returned.

When the machine is first powered on, the configuration software must scan the various buses in the system (PCI and others) to determine what devices exist and what configuration requirements they have. This process is commonly referred to using any of the terms below:

- Scanning the bus
- Walking the bus

- Probing the bus
- The discovery process
- Bus enumeration

The program that performs the PCI bus scan is called the as the "bus enumerator". The configuration software then proceeds to read from the device's other configuration registers to determine the resources required by the device.

Each PCI add in connector implements two card present bits referred to as PRSNT1# and PRSNT2#. If a card connector is unoccupied a value of 11b is read from its two card present signals. Any other value indicates that a card is installed in the connector.

When the configuration software has determined that a card connector is occupied, it can determine the card type by reading from its vendor and device ID configuration registers as shown in Figure 3.2 and Figure 3.3.

Once a device is detected the configuration program reads from its configuration header registers to determine its resource requirements. The configuration program can then write the appropriate values to these same registers to allocate non-conflicting resources to the device.

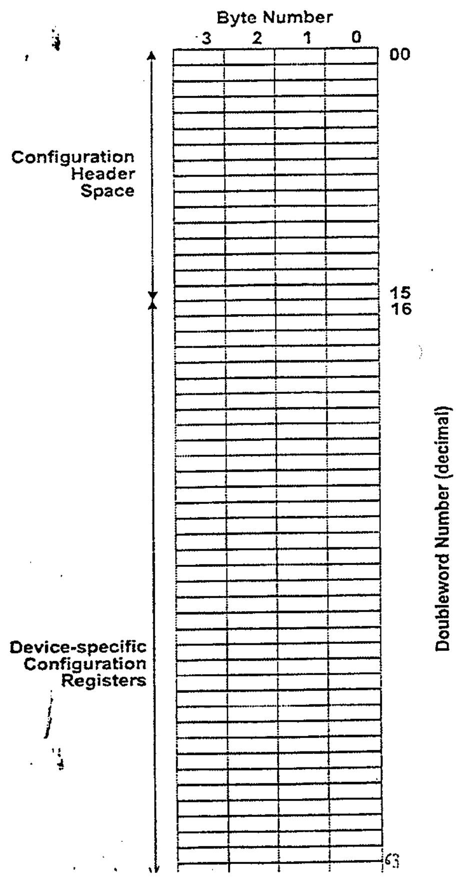
Byte Number
3
2
1
0
00
Configuration Header Space
15
16
Doubleword Number (decimal)
Device-specific Configuration Registers
63

Figure 3.2 PCI Configuration address space from Ref. [10]

Once the resources have been allocated to the device the configuration program writes the appropriate value to its command register to enable the device for normal operation.

In order to facilitate this process each PCI function must implement a base set of configuration registers defined by the PCI specification. The configuration software reads a subset of a device's configuration registers in order to determine the presence of the function and its type. Having determined the presence of the device the software then accesses the function's other configuration registers to determine how many blocks of memory and/or I/O space the device requires. It then programs the device's memory and/or I/O address decoders in order to respond to memory and/or I/O address ranges that are guaranteed to be mutually exclusive from those assigned to other system devices.

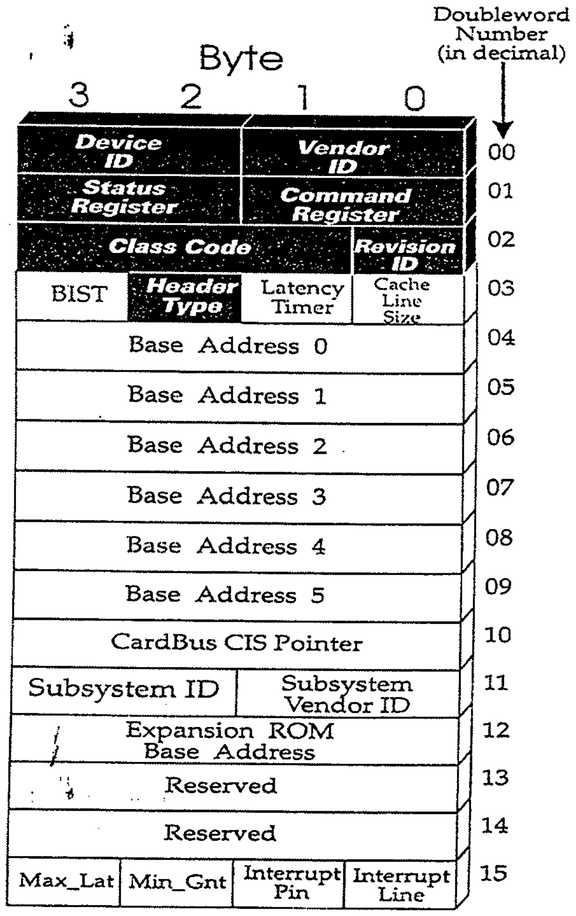

Figure 3.3 PCI Configuration header details from Ref. [10]

If the function indicates usage of a PCI interrupt request pin via one of its configuration registers, the configuration software programs it with routing information indicating what system interrupt request line the function's PCI interrupt request pin is routed to by the system.

If the device has bus mastering capability the configuration software can read two of its configuration registers to determine how often it requires access to the PCI bus (arbitration priority need) and how long it would like to maintain ownership in order to achieve adequate throughput. The system configuration software can utilize this information to program the bus master's Latency Timer register and the PCI bus arbiter to provide the optimal PCI bus utilization.

A PCI device that contains only one function is referred to as a single function device. A PCI device that contains more than one function is referred to as a multi function device. A bit in one of a function's configuration registers defines whether the package contains one function or more.

Intel x86 and Power PC 60x processors possess the ability to address two distinct address spaces, I/O and memory. PCI bus masters use PCI I/O and memory transactions to access PCI I/O and memory locations respectively. In addition, a third access type, configuration access, is used to access a device's configuration registers. A device's and its function's configuration registers must be initialized at startup time to configure the function to respond to memory and/or I/O address ranges assigned to it by the configuration software.

The PCI memory and I/O space is 4GB in size. PCI configuration space is divided into a separate, dedicated configuration address space for each function contained within a PCI device (in a chip or on a card). The first 16 double word (dword) part of a function's configuration space is referred to as the function's configuration header space. Three header types are currently defined. These are:

- Header type zero (for all devices other than PCI-to-PCI bridges)
- Header type one (for PCI-to-PCI bridges)
- Header type two (for Card Bus bridges)

The system designer must provide a mechanism that the Host/PCI Bridge will use to convert processor-initiated accesses with certain pre-defined memory or I/O addresses into configuration accesses on the PCI bus.

According to the 2.2 PCI spec. Every device other than host bus bridges must implement configuration address space. Host bus bridges may optionally implement configuration address space. If the host /PCI bridge doesn't implement its configuration registers in PCI configuration space, its configuration registers may be implemented in either I/O or memory-mapped I/O space. Memory mapped I/O space is generally better because x86 I/O space is small (64 KB total) and crowded with other configuration information for the PC system.

Initially the BIOS performs device configuration and once a plug and play OS, such as Windows 98, has been booted, device management control is passed to it.

The programmer must supply the following information to the Host/PCI Bridge when performing a configuration read or write.

- Target PCI bus.
- Target PCI device on the bus
- Target PCI function within the device
- Target double word within the function's configuration space.
- Target byte within the double word.
- The configuration mechanism utilizes two 32-bit I/O ports located at addresses 0CF8h and 0CFCh. These two ports are:

– 32-bit configuration address port (occupies I/O addresses from 0CF8h through 0CFBh) as it is shown in Figure 3.4

– 32-bit configuration data port (occupies I/O addresses from 0CFCh through 0CFFh)

Accessing one of the PCI function's configuration registers is a two-step process:

– Write the target bus number, device number, function number and double word number to the configuration address port and set the enable bit in it to one, indicating that the configuration process is enabled.

– Perform a four byte I/O read from or a write to the configuration data port

In response the Host/PCI Bridge compares the specified target bus to the range of buses that exist on the other side of the bridge and if the target bus resides beyond the bridge, it initiates a PCI configuration read or write.

Any 8 or 16-bit access within this I/O double word is passed directly on to the PCI bus as an 8 or 16-bit PCI I/O access.

The information written to the configuration address port is latched by the host /PCI bridge. If bit 31 is set to one and the target bus number compares to the bridge's PCI bus number register, the bridge is enabled to convert a subsequent processor access, targeting its configuration data port into a PCI configuration access (see Figure 3.4). The processor then initiates a one-byte, two-byte, or four byte I/O read from or a write transaction to the configuration data port at 0CFCh. This stimulates the bridge to arbitrate for ownership of the PCI bus and then to perform a configuration read or write. It will be a type-0 configuration transaction if the target bus is PCI bus 0, or a type-1 configuration transaction if the target bus is further out in the bus hierarchy beyond bus 0. We will talk about these configuration methods later.

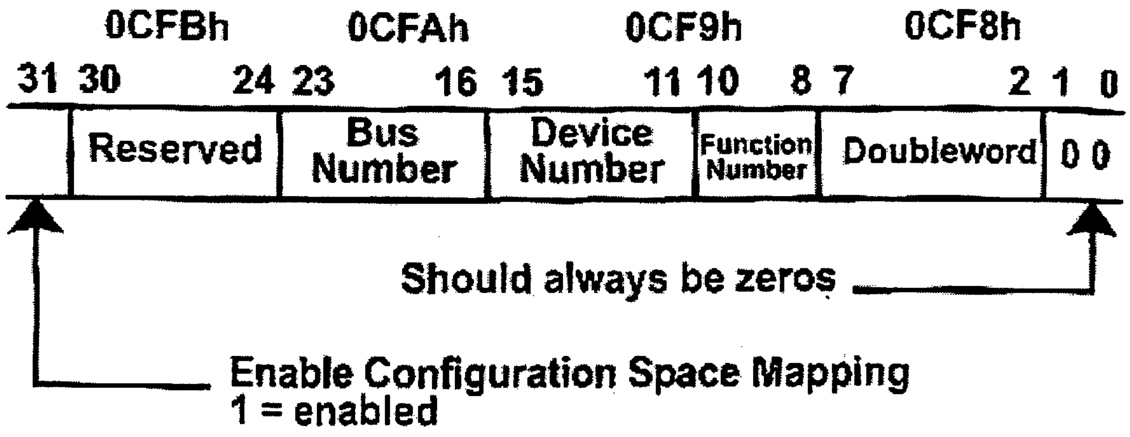

Figure 3.4 PCI Configuration Address port at 0CF8h from Ref. [10]

When the operating system begins to read loadable device drivers into memory the device driver's initialization code calls the BIOS. The BIOS scans the PCI bus by reading the vendor and device IDs from every device and looking for a match. When a match is encountered the BIOS returns the device number to the driver along with the PCI bus number and the function number, which identifies one of eight functions within the target physical device. This way the driver has the information to reach the configuration registers of the PCI device.

As a summary, PCI devices can be automatically configured without any intervention by the end user. In addition, the OS can identify the driver associated with the device and load it into memory.

D. PCI CONFIGURATION ACCESS METHODS

1. TYPE 0 Configuration Access:

When devices that reside on a PCI bus detect a type-0 configuration in progress this informs them that one of them is the target device. When devices that reside on a PCI bus (other than PCI-to-PCI bridges) detect a type-1 configuration access in progress they ignore the transaction.

The Host/PCI Bridge latches the information written to the configuration address port. If bit 31 is set to one and the target bus or subordinate bus number is equal to or less than the bridge's PCI bus number register, the bridge is enabled to convert a subsequent processor access targeting its configuration data port into a PCI configuration access. The processor then initiates a one-byte, two-byte or four-byte I/O read or write-transaction to the configuration data port at 0CFCh. This stimulates the bridge to arbitrate for ownership of the PCI bus and then to perform a configuration read or configuration write. It will be a type-0 configuration transaction if the target bus is PCI bus-0, or a type-1 configuration transaction if the target bus is further out in the bus hierarchy beyond bus-0.

a. TYPE 0 Configuration Transaction

(1) Address phase: During any PCI transaction, all PCI devices on the bus latch the following information at the end of the address phase:

- The contents of the AD bus:

<u>For type 0:</u>

Target function, configuration double word and 00b as shown in Figure 3.5.

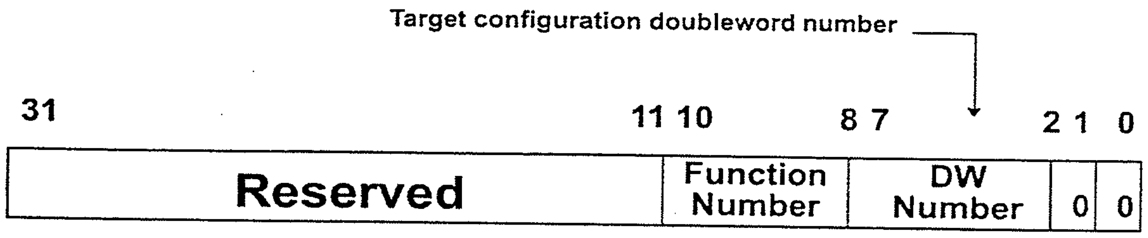

Figure 3.5 Type 0 configuration address register contents from Ref. [10]

<u>For type 1:</u>

– Target bus number, device number, function number, dword number and 01b as shown in Figure 3.6.

– The state of the FRAME# signal, which indicates the presence of a valid start address and transaction type on the bus.

– The state of the IDSEL signal, which is an input signal to the PCI device and used as chip select, only for type-0 configuration transaction.

– The command on the command or byte enable bus, C /BE# [3:0], which defines the type of transaction (configuration read or configuration write).

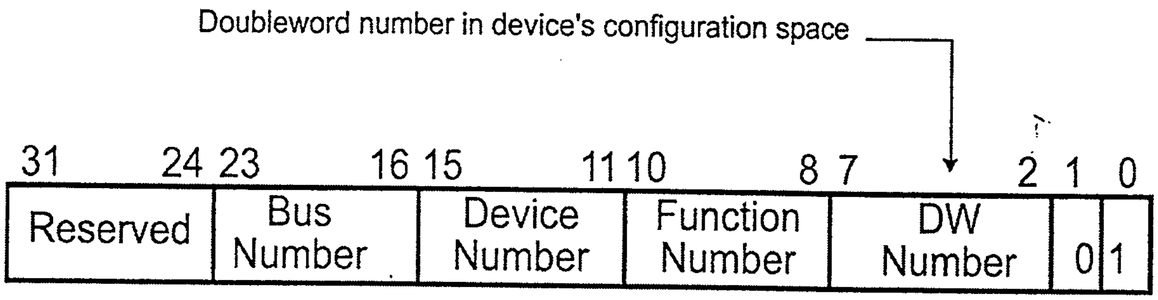

Figure 3.6 Type 1 configuration address register contents from Ref. [10]

The PCI device that samples its IDSEL asserted is the target device. If address bits AD [1:0] are 00b, this indicates that this is a type 0 transaction targeting one of the devices on this bus, note Figure 3.5. The AD [7:2] indicates the target configuration double word. The AD [10:8] indicates the target function within the physical device selected by the IDSEL signal. The ADs [31:11] are reserved and must not be interpreted by any devices.

The target device number specified in bits [15:11] in the configuration address port are decoded within the bridge and the decoder asserts the IDSEL output signal during the transaction's address phase, note Figure 3.7. If there is no device on its secondary bus no DEVSEL#, which is a signal asserted by the target device to inform the initiator that the target has decoded its address, will be asserted and the bridge will cease the transaction with a master abort. Theoretically 32 devices can be implemented on a PCI bus but in reality this number cannot exceed 10 because of the electrical load limitations.

There are two methods in implementing the IDSEL signal routing. In the first method the IDSELs are routed over unused AD lines, see Figure 3.8 and Figure 3.9 for different implementations. The second method implements separate IDSEL output pins and traces.

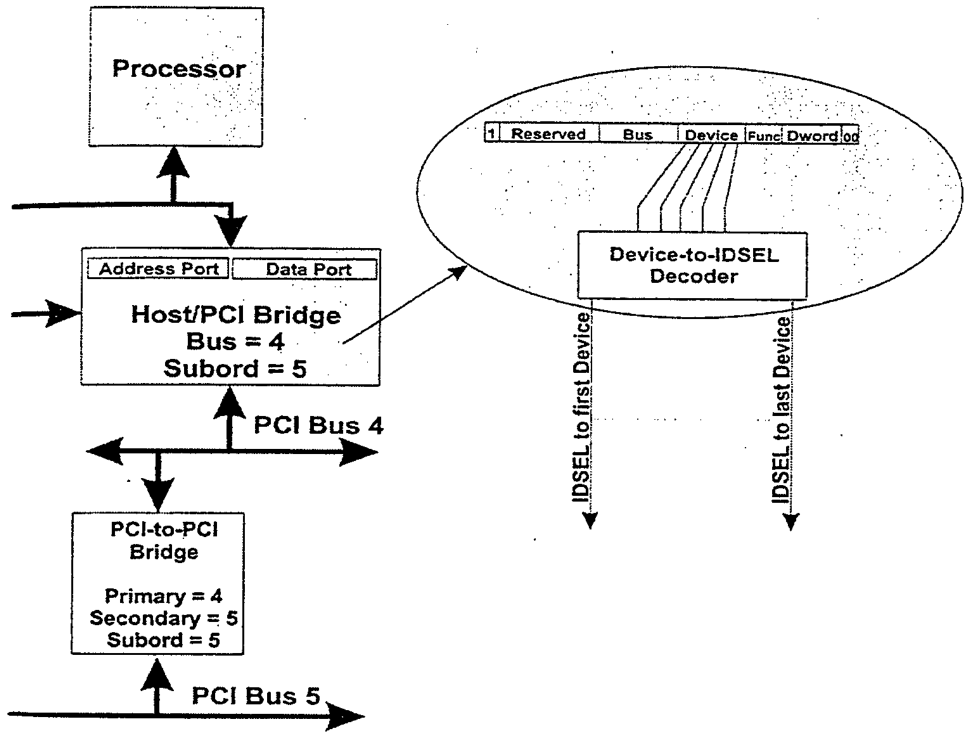

Figure 3.7 Host/PCI bridges device decoder from Ref. [10]

PCI-to-PCI bridges must use the first method. In the first method the upper 21 address lines, which aren't used during the address phase in a type 0 access are used for the IDSEL signal routing.

In general the signal mapping recommended is given below.

- IDSEL device 0 → AD16
- IDSEL device 1 → AD17
- IDSEL device 2 → AD18
- ....
- IDSEL device 15 → AD31

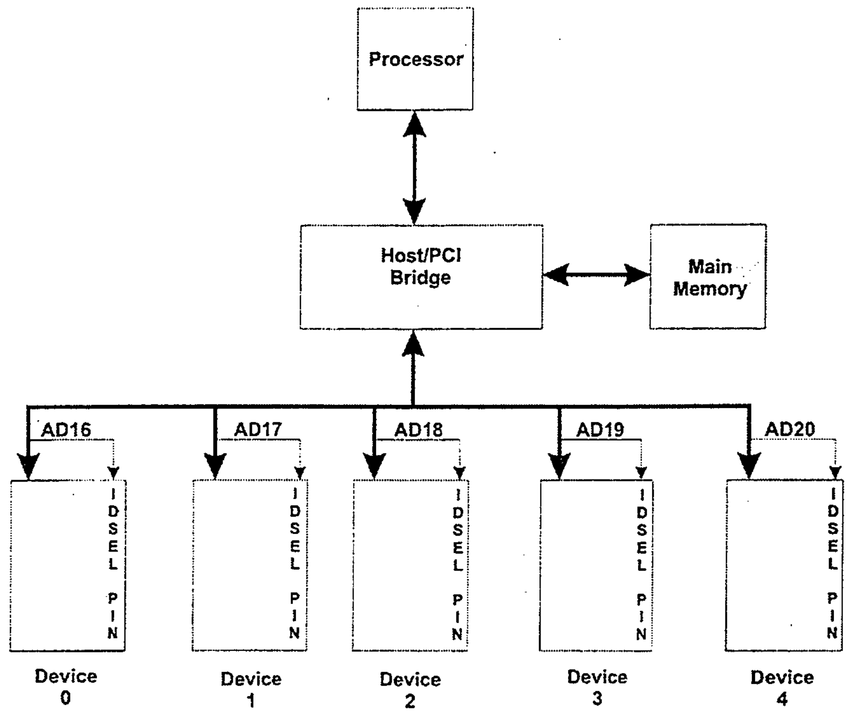

Figure 3.8 Direct connection of IDSEL pins to AD lines from Ref. [10]

In the second method, after the host/PCI bridge decodes bits [15:11] in the configuration address port, it will assert the target physical devices IDSEL output signal line. This method requires extra IDSEL pins on the bridge and a separate point-to-point IDSEL line (trace) on the system motherboard between the bridge and each PCI device or connector. This is not a preferred solution in real life. The first method is mostly preferred in real life implementations.

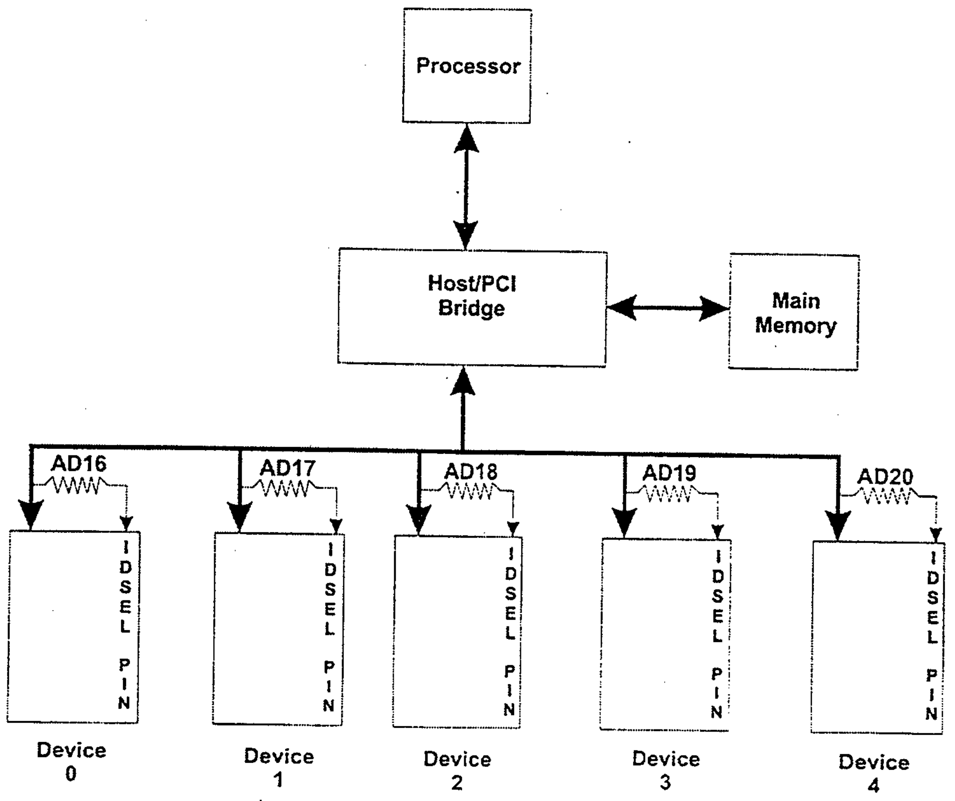

Figure 3.9 Resistive-Coupling of IDSEL pins to AD lines from Ref. [10]

The host/PCI bridge initiates the type 0 configuration transaction by driving out the target function and double word number on the address bus with AD[1:0] set to 00b to indicate that this is a type 0 configuration transaction. It also outputs its internal device decoder's IDSEL output signals onto the upper AD lines. The configuration read or write is driven onto C/BE#[3:0]. No devices will pay attention to the transaction until FRAME# is asserted.

(2) Data phase: As the data phase is entered the bridge sets the C/BE#[3:0] to indicate which bytes within the currently addressed double word will be transferred. The bridge gets this information from the processor's access to the bridges configuration data port .

- C/BE#[0] asserted → one byte (read or write)
- C/BE#[1:0] asserted → two byte (read or write)
- C/BE#[3:0] asserted → four byte (read or write)

When a device detects that its IDSEL pin was asserted at the end of the address phase, it must determine whether or not to claim the transaction. How it does this depends on whether it is a single function or a multi-function device. According to the PCI version 2.2 specs:

(a) Single-function device:

- Decodes the function number and only asserts DEVSEL# for function zero
- Or may respond to all function numbers other than zero by not asserting DEVSEL and allowing the transaction to terminate via a master abort.

(b) Multi-function device:

- must implement a function decoder
- must decode the function number delivered on AD[10:8] during the address phase.
- If the target function is implemented, the device asserts DEVSEL# and claims the transaction
- Otherwise it ignores the transaction.

2. TYPE 1 Configuration Transaction

When a bridge initiates a configuration access on a PCI bus, it places the configuration address information on the AD bus and the configuration command on the C/BE bus. During the address phase of a type-1 configuration access, the information on the AD bus is formatted as follows:

– AD [1:0] contain a 01b. Type 1 configuration access.

– AD [7:2] identifies one of 64 configuration double words within the target device's configuration space.

– AD [10:8] identifies one of eight device dependent functions within the target physical device.

– AD [15:11] identifies one of the 32 physical devices. This field is used by the bridge for the selection of which device's IDSEL line to assert.

– AD [23:16] identifies one of the 256 PCI buses in the system

– AD [31:24] are reserved and cleared to zero.

The configuration-read or write message is presented on the C/BE bus during the address phase. During a type 1 configuration access, PCI devices ignore the state of their IDSEL inputs.

When any PCI-to-PCI bridge latches a type-1 configuration access on its primary side, it must determine which of the following actions to take:

– If the bus number field on the AD bus doesn't match the number of its secondary bus or its subordinate buses then it ignores the access.

– If the bus number field matches the bus number of its secondary bus it will claim and pass the configuration access onto its secondary bus as a *Type 0 configuration access*.

– AD [1:0] on the secondary bus are set to 00b,

– AD [10:2] are passed to its secondary AD bus.

– The device number field is decoded within the bridge to select one of the IDSEL lines to assert on the secondary bus.

– The configuration command is passed from the primary to the secondary C/BE bus.

– If the bus number is not equal to the secondary bus number but is within the subordinate bus number range (small or equal) the bridge claims and passes the access through as a *Type 1 configuration access.*

– AD [31:0] are passed to its secondary AD bus.

– The configuration command is passed from the primary to the secondary C/BE bus.

When the target device does not exist DEVSEL# is not asserted by any PCI function. If this is the case, the Host/PCI Bridge Master Aborts the transaction and sets the master abort bit in its configuration status register. If Master Abort happens on a read, the bridge will return all ones to the processor, as the read data (if vendor ID read is FFFFh no device exists). If the Master Abort happens on a write, the bridge acts as if the write completed OK.

In this chapter we went over the PC boot-up process and the PCI protocol. In the next chapter we will use this information to propose solutions for object reuse control of storage areas, each of which were analyzed in the previous chapter.

IV. OBJECT REUSE CONTROL ANALYSIS

In an MLS system, resources are shared under the control of the system. In the client PC the storage objects could become an information transfer channel between disjoint users as the system reassigns the objects to users. If the object access is not controlled appropriately objects can be accessed by system calls, through use of memory scanners or debugging tools. This is a problem because, when an object is deleted by the operating system, only the pointer to the location of the object is erased, not the object itself, leaving the information in the object vulnerable to unauthorized accesses. In this chapter we are going to discuss solutions for the object reuse control of PC main memory, PC CPU cache, Accelerated Graphics Port memory and Real Time Clock memory.

A. OBJECT REUSE FOR MAIN MEMORY

The PC main memory (DRAM) is one of the main storage objects in the client computer. We want object reuse control over the main memory to prevent any sensitive information leakage between two sessions.

In chapter-II we analyzed the motherboard and the components on it. Now we have an understanding about the functioning of these devices on the motherboard. In this chapter we will investigate possible ways to erase PC main memory to fulfill object reuse requirements. This can be done through software or by using specially designed hardware. We are going to take look at these options and decide on the best one in terms of practicality, cost effectiveness, and easy installation and administration.

1. Software Based Object Reuse Controls

In this section, software based object reuse control is intended to cover all of the procedures used for erasing or overwriting the main memory by means of code executed by the TCBE CPU.

a. Selectively Overwriting the Parts of the Memory from the TCBE

(1) Concept of technique: Main memory can be selectively overwritten using a software-controlled mechanism. If the TCBE is able to take control of the bus as the bus master, it can write random bit patterns over the predefined address regions of memory. As seen in Figure 4.1, the TCBE will overwrite the shaded areas and leave the areas defined as configuration data, unchanged. This is accomplished by using a software routine, to be initiated by the TCBE, when memory purges are required. This routine will reside in the memory of the TCBE since the memory unit will be battery-backed.

Blocks of main memory can be overwritten by memory access from the TCBE CPU or by establishing dynamic memory access (DMA), via the DMA controller. In the DMA access, the TCBE provides the starting address of the block, the amount of the data to be overwritten and the data to be copied. When this type of access is initiated the blocks of memory will be overwritten by the DMA controller in the TCBE without the participation of the TCBE CPU. The difference between DMA access and direct CPU memory access is that DMA provides block access while direct CPU access provides line by line access.

The configuration data is the information gathered about the PC system and its components for the system to operate most efficiently. The system saves this information into the main memory to provide fast and easy access when this information is needed.

In this method we want the configuration data to be preserved so that the computer doesn't have to spend extra time to gather all the configuration information.

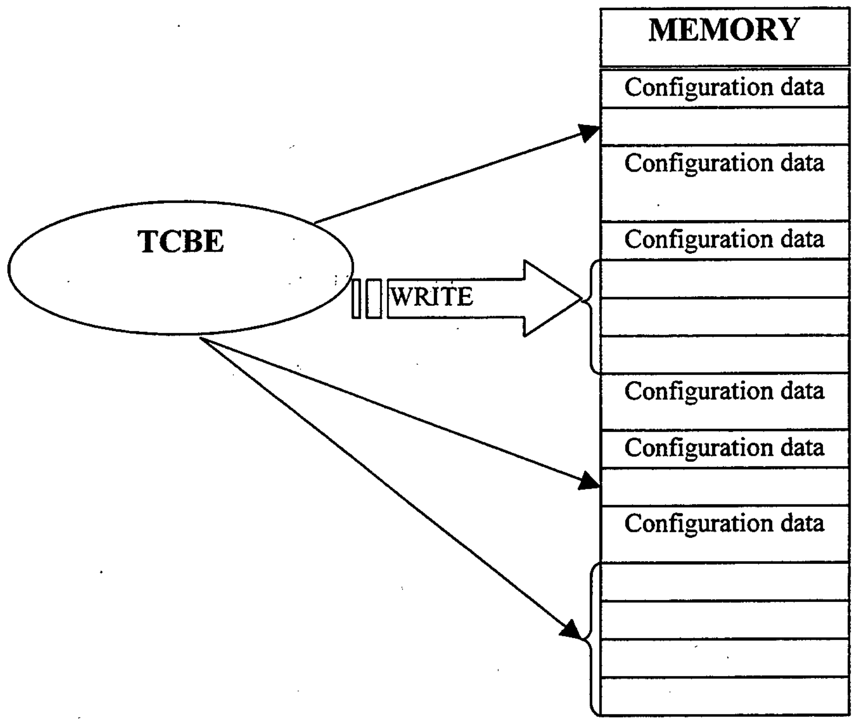

Figure 4.1 selectively overwriting the memory

(2) Advantages and disadvantages: The use of the software oriented controlling will bring flexibility to the design of the TCBE. It is easy to install and configure a software-based control system.

Since the configuration data preserved in the memory by the system is predefined, the overwriting process will be applied only to memory regions, which do not contain the configuration information. This way the overwriting process won't affect the configuration regions and the configuration data will be preserved.

No extra hardware is needed. The only hardware required is that already planned for the TCBE: memory and its processor. This will keep the TCBE hardware design simple and cost effective.

There are several disadvantages to this approach. The first is that writing over the memory is a time consuming process. It takes many bus cycles to overwrite large portions of memory. Consider a PCI bus operating with a bus speed of 100 MHz. Even when the latency caused by the memory components and the PCI protocol is neglected, assume that every clock cycle we can write 4 bytes (32 bit data bus) each clock cycle, which means zero latency, into a memory block of 128Mb. This means that we can write over the whole memory in 0.32 seconds. Which is a big amount of time considering with the usual PC operating speed. The total time for the overwriting process will even get close to half second if we consider the approximate additional effect of latency caused by the hardware components such as arbitration, wait states, repeat cycles, buffering, etc.

Another disadvantage is the possibility that the configuration data space might be used by malicious application code to store information. So that sensitive information can be carried from one session to another session.

Bus arbitration presents another problem. Bus ownership may not be continuous during the memory overwrite process. Guaranteed bus ownership (atomic bus ownership) is needed. Atomic bus ownership doesn't seem to be very possible since there are other hardware-oriented factors such as the non-maskable interrupts. For example, if a non-maskable interrupt (such as a reset signal) is issued, it has to be serviced by the system CPU. This will cause all bus transactions to cease. Another reason for losing bus ownership is expiration of the arbitration time, thereby bus ownership passes to another bus master. This puts control of object reuse for memory in jeopardy.

Also after the overwriting process has completed, no untrusted application should be allowed to run before the new session is established. The untrusted application may access the main memory and write information in it making the whole overwriting process useless.

b. Overwriting the Whole Memory from the TCBE

(1) Concept of technique: By using a software-only mechanism the whole main memory can be overwritten. The overwriting software will reside in the memory of the TCBE. When the TCBE acquires bus ownership, it will write a random bit pattern into memory. This process can use either a direct memory access or DMA-oriented I/O access.

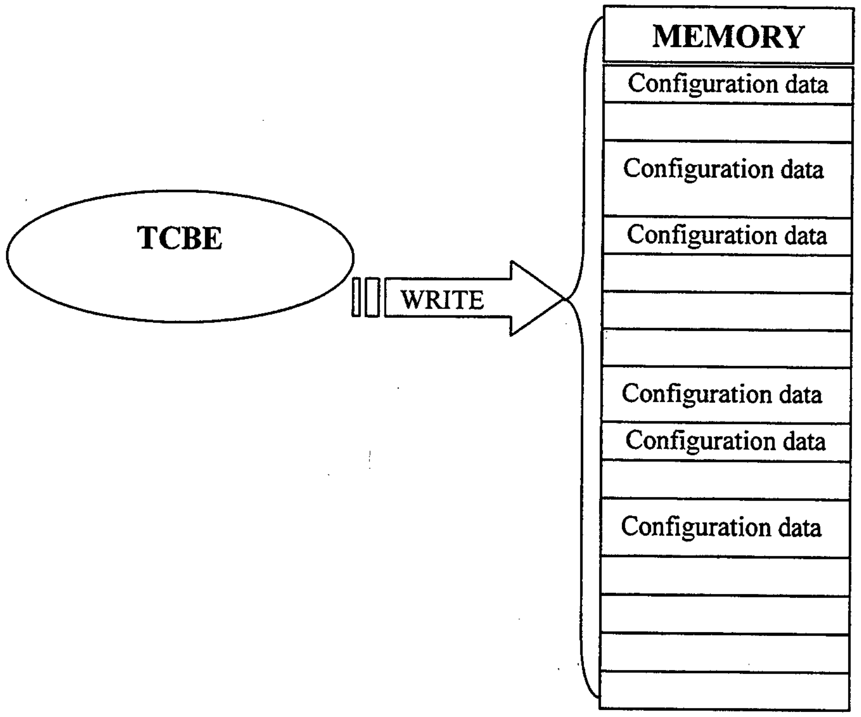

Figure 4.2 overwriting the whole memory

(2) Advantages and disadvantages: In this approach no extra hardware is required. This will keep the TCBE hardware design simple and cost effective.

It is easy to install and configure a software based control system. The use of software oriented control will bring flexibility to the design of the TCBE.

Overwriting all of memory will eliminate the problem of malicious code trying to pass information between sessions by using the configuration space mapped into the memory.

A disadvantage of this method is that the configuration data will be lost since the whole memory is overwritten. This will require another configuration process by the system at the next session, which will consume time.

Also, overwriting memory is time consuming, as we investigated in the previous section, and it will cause delays between session changes.

The other problem is the bus arbitration problem. Bus ownership may not be continuous during the memory overwrite process. Guaranteed (atomic) bus ownership is needed. Atomic bus ownership doesn't seem to be very possible since there are other hardware-oriented factors such as the non-maskable interrupts. If a non-maskable interrupt (such as a reset signal) is issued, it has to be serviced by the system CPU. This will cause all the bus transactions to cease. Another case for losing bus ownership is when the arbitration time expires thereby passing bus ownership to another bus master.

The overwritten area should be write-protected to prevent a secondary write, which can be initiated by malicious software to write sensitive information back into memory again, during or after the overwriting process. This process is like picking up marbles from the carpet before the vacuum cleaner sweeps the carpet and dropping them on the floor again following right behind the vacuum cleaner. Also after the overwriting process no untrusted application is allowed to run before the new session is established.

c. *Memory Controller Reconfiguration by TCBE*

(1) Concept of technique: DRAM memory cells are made of capacitors. Each cell defines one bit with a logical value of one or zero. The logical value of a memory cell, whether it is a logical zero or logical one, is defined by measuring the voltage level in the cell. The memory cell cannot hold the voltage at a constant level for a

sustained period of time. Eventually, the voltage level of the cell drops down. For consistent data storage, the memory cell needs to be periodically energized (refreshed) to preserve its voltage level. If DRAM cells are not refreshed every 64 milliseconds or sooner, the data in this type of memory will be lost. Software may be used to reconfigure the memory controller to regulate the DRAM refresh rate. Here the TCBE would use the PCI bus communication protocols to control and communicate with the memory (DRAM) controller device within the 82443BX Host/PCI bridge chipset.

In our prototype we have a single Host/PCI bridge. We need to configure the memory (DRAM) controller on the chipset by writing the appropriate configuration value to the configuration address port and the configuration data port, which will be latched by the Host/PCI Bridge. After this, the bridge will arbitrate the PCI bus and start a configuration write/read process. When the configuration address and data are written to the configuration ports by the TCBE, the 82443BX Host/PCI Bridge is triggered and it sets the appropriate configuration registers of the PCI device, in this case those of the DRAM controller. If the PCI device is on the system PCI bus, the bridge will arbitrate the bus to configure the PCI device.

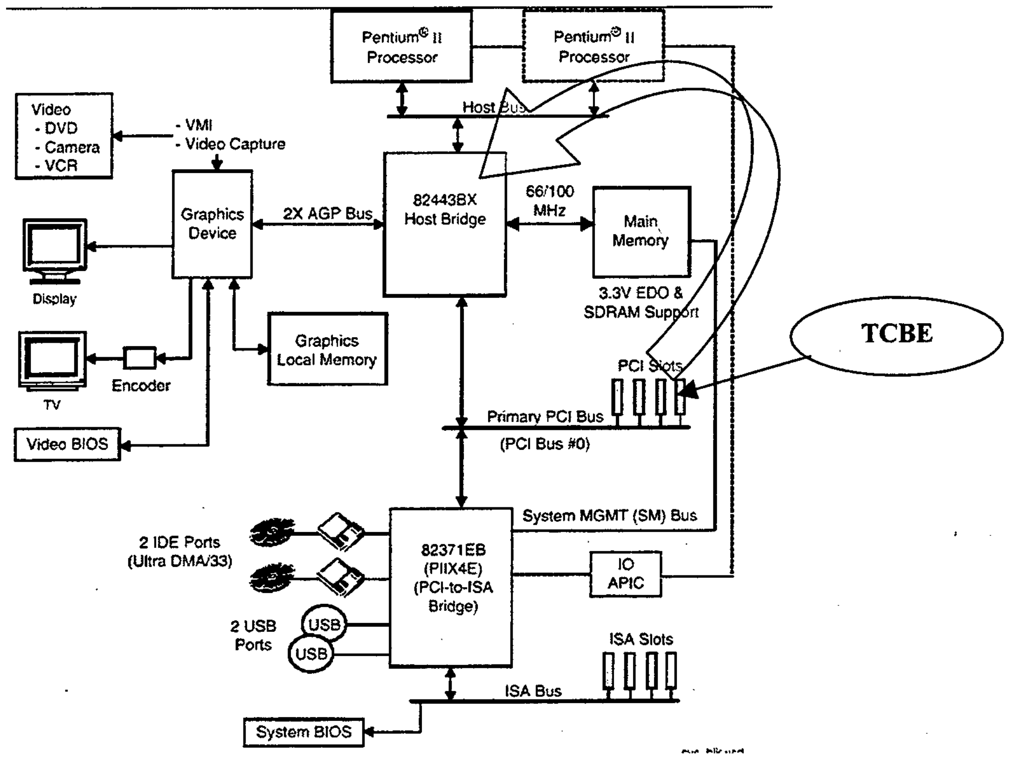

Figure 4.3 clearing the memory via DRAM refresh rate configuration, after Ref. [4]

By writing appropriate data to the configuration registers of the 82443BX host–bridge, the DRAM controller can be configured so that it will stop the DRAM refresh process and we can clear the memory.

The 443BX chip set uses the configuration access mechanism by utilizing the CONFADD register and CONFDATA register. To reference a configuration register, a double word I/O write-cycle is used to place the required information into the CONFADD register. After this process any read or write to the CONFDATA register leads to a double word PCI configuration access to the PCI device's configuration address space, which is 64 double words (one word is 2 bytes) in size. [Ref. 4]

For the 443BX chipset, the I/O address for the CONFADD register is 0CF8h and for the CONFDATA register it is 0CFCh. To configure the DRAM controller for no refreshing (zero refresh), the value to be written into the configuration address port will be 0x80000021 and for the configuration data port it will be 0x00000000. The configuration process has been explained in the previous chapter.

(2) Advantages and disadvantages: This method of erasure has the flexibility of the software oriented system approach. It provides easy administration and configuration of the system. No extra hardware is required and this will help to keep the TCBE design simple and cost effective. This also indirectly provides easy hardware installation to the client.

Clearing the memory is faster than sequentially overwriting it. The DRAM main memory needs to be refreshed every 64 milliseconds and if the DRAM controller doesn't provide this minimum refresh rate the data in the DRAM cells will eventually be lost. The writing process takes a lot longer than this. As it is calculated before, the memory overwriting process for a 128Mb. memory over a 100MHz. PCI bus takes at least 0.32 seconds. By controlling the refresh rate of DRAM, the clearing process is five times faster (0.32/0.064 = 5) than the overwriting process.

One of the disadvantages of this method is the bus arbitration problem. Guaranteed bus ownership (atomic bus ownership) is needed until the zero refresh configuration process is completed. Also, the continuity of the zero refresh configuration needs to be preserved until the whole system is cleared and checksumed (the next session). The zero-refresh configuration must be held at least 64 milliseconds. Otherwise, if the zero-refresh configuration is changed back to the refresh mode before the 64 milliseconds elapses, the memory cannot be cleared.

2. Hardware Based Object Reuse Control

a. Clearing the Memory by Making Direct Hardware Connections to the Memory from the TCBE

(1) Concept of technique: The DRAM memory needs to be refreshed to retain its data in the memory cells. The DRAM controller on the 82443 BX host/PCI chipset controls the refresh process. For the refresh process to occur certain signals must be supplied to the DIMMs. These signals are fed into the predefined pins of the DIMM card.

Some of these signals are active high (3.3 volts) and some of them are active low (0 volts), which means that the state of these signals provide the refresh configuration information to the DRAM memory. The state of the refresh configuration depends on the combination of assertion/deassertion of these signals. If we can block these signals in a way to force the refresh rate to be zero we can prevent the refresh process from happening causing the DRAM to be cleared. This way we can erase the memory to provide object reuse control between sessions.

This requires a hardware mechanism. The basic functionality of the hardware mechanism is to provide the required signals to the required pins. The signals will be controlled by the TCBE. The derivation of the required signals to design the logic circuit for providing the zero-refresh rate is given in Appendix A.

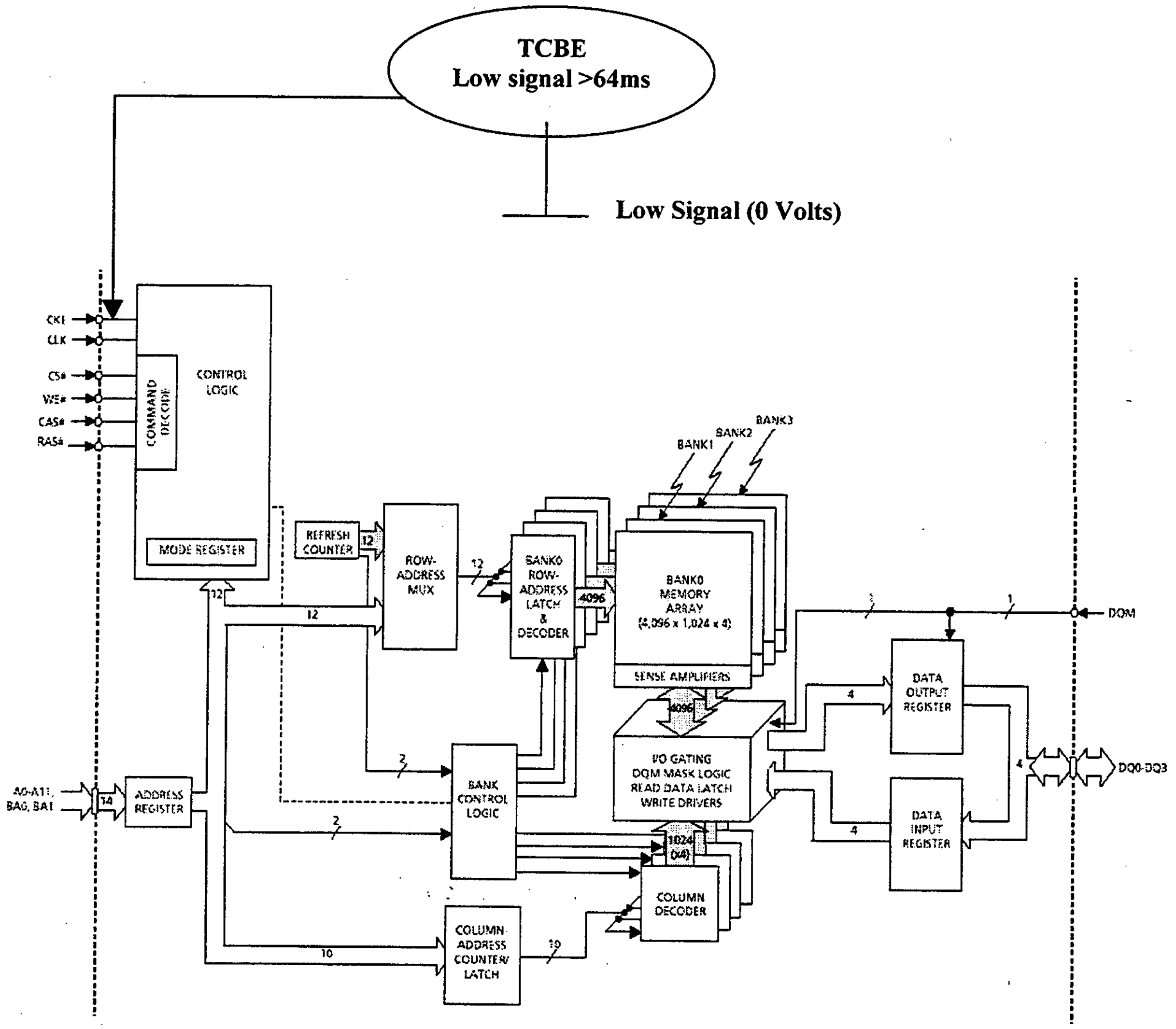

Figure 4.4 clearing the memory by using a direct hardware connection, after Ref. [23]

(2) Advantages and disadvantages: By making direct hardware connections to the DIMMs we bypass all the rules enforced by the PCI bus architecture and other system rules which limits our control over the memory. This method ensures a guaranteed connection to the memory. This process is direct and is completely controlled by the TCBE. The TCBE doesn't need to make any request to the system to take over control. It simply overrides the system memory control signals. There is no other way to avoid this process without direct physical access.

This process is very fast because we only need to supply the zero-refresh signal for 64ms. Memory will loose all the information in it. It is faster than overwriting all of memory. Since a hardware device residing on the TCBE originates the mechanism and since the connections are not routed by using the system bus there is no possibility of malicious software interference. Control remains within the TCB perimeter at all times.

A disadvantage of this approach is the increased hardware complexity of the TCBE. For every different hardware configuration involving the chip set and the DIMM structure, a different hardware design will be required to accomplish the same functionality. To provide easy installation a standard chipset and DIMM configuration would be needed.

For different chipsets, a more general hardware design can be built to provide compatibility for different hardware configurations, since the functionality is just a matter of providing the appropriate signals to the DIMM to cause the termination of the refresh process.

The increase in the complexity of the TCBE hardware design resulting from the introduction of the extra hardware device will increase its cost.

b. TCBE Hardware-Controlled Partial Memory Clearing

(1) Concept of technique: Memory can be erased by making direct hardware connections from the TCBE to the DIMM blocks. The refresh rate can be controlled by the appropriate signals provided to the DIMM module from the TCBE by intercepting the signals from the DRAM controller. A partial erasing process can be

achieved if the DIMM module-refresh process can be controlled separately. This is possible if the refresh configuration allows existing regions of a certain size in the memory module to be refreshed independently. This way clearing a desired memory block would be possible.

This process may not be realized currently since the DIMM technology doesn't provide the separate refresh controlling of the memory modules. The ability to control the refresh rates of individual memory blocks will also improve the object reuse control level on the memory by clearing the memory blocks separately. This way any desired memory block or series of blocks can be erased providing the configuration data to be preserved while the rest of the memory is wiped out. This can be accomplished by using the same methodology in the previous concept of technique.

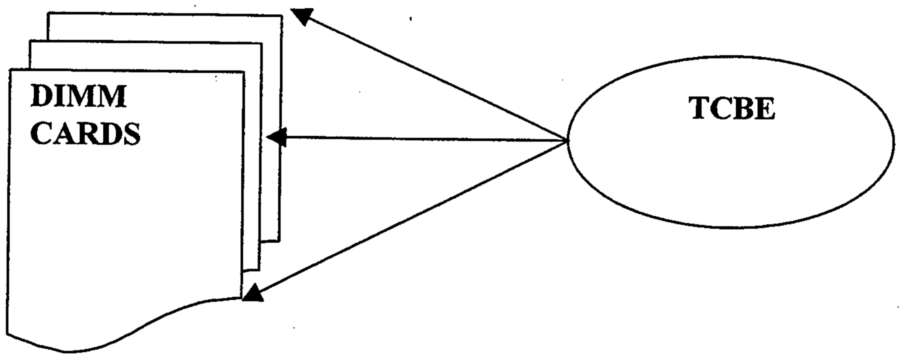

Figure 4.5 partial clearing of the memory

(2) Advantages and disadvantages: By making direct hardware connections to the DIMMs we bypass all the rules enforced by the PCI bus architecture and other system rules which limits our control over the memory. This ensures a direct connection from the TCBE to memory so that we can be sure that the memory is erased.

Another advantage of this approach is that the configuration data written to memory can be preserved, since we can selectively erase memory. This method is faster than the one accomplished by software.

Besides having a more complex hardware design, a disadvantage is the problem of a malicious code hiding the sensitive data into the configuration address space. Malicious code may hide sensitive information in the system memory space reserved for the configuration data at any time. Since the configuration data is preserved in this method, sensitive data may be passed over to the next session via the system memory. Also, we need to ensure that the cleared memory is not going to be used again until the next session starts.

c. *Powering off the Whole PC and Keeping the Power on the TCBE*

(1) Concept of technique: We can turn off the power supply of the whole PC so that all volatile storage including the main memory is erased. At the same time we must still provide power to the TCBE so that it can maintain control. This can be accomplished in two ways. Extra circuitry can be added to manage the power from a single source, providing the required power to the TCBE and to the PC, as in Figure 4-6, or the TCBE can be supplied with its own power supply, as in Figure 4-7.

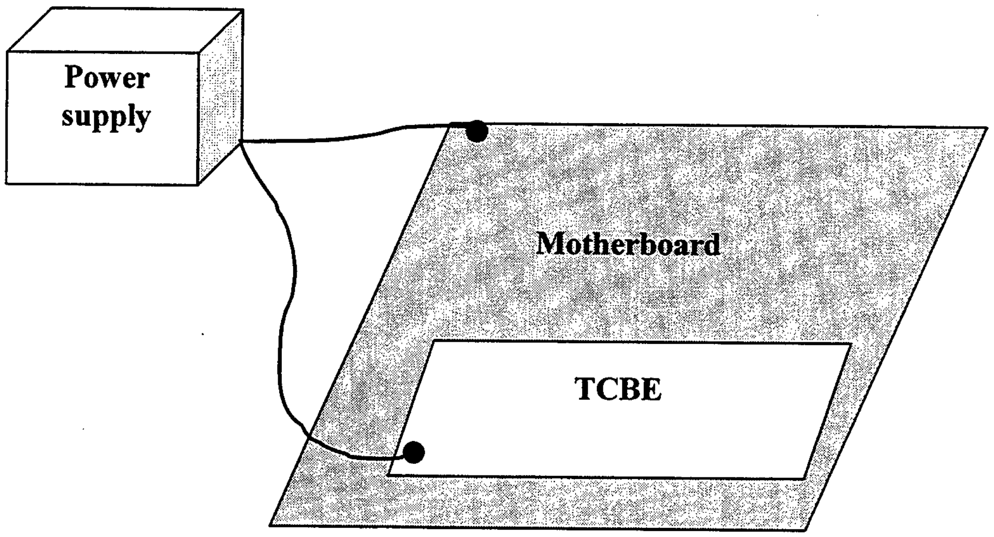

Figure 4.6 TCBE sharing the same power supply

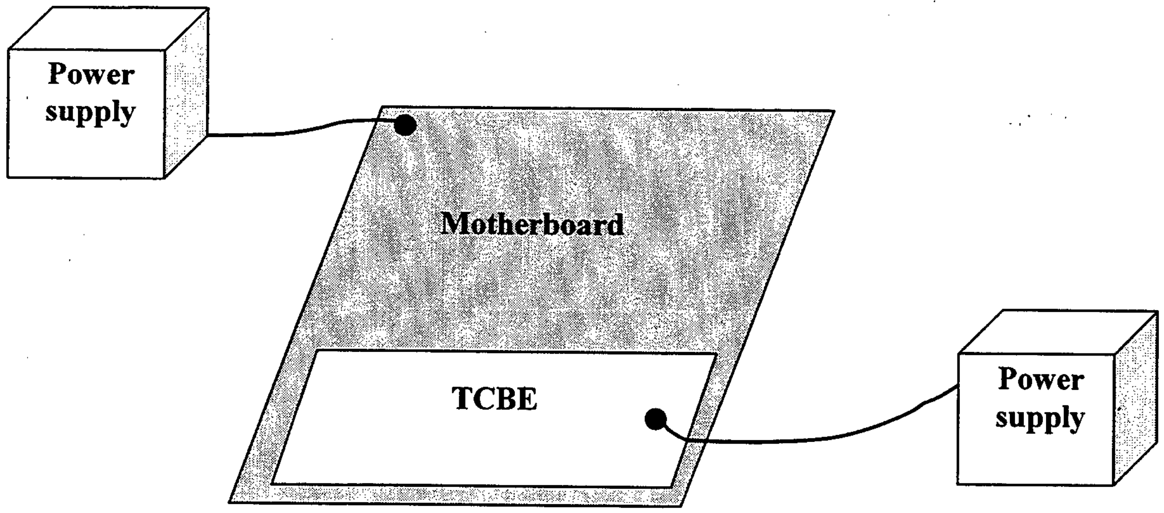

Figure 4.7 TCBE with its own power supply

(2) Advantages and disadvantages: Since the whole PC will loose the power all the volatile storage areas will be cleared. This will ensure that memory is erased and no sensitive data is retained in main memory between two sessions.

On the other hand, this process will be time consuming since we have to power up the whole computer again. We have to wait for all the devices to come alive.

Extra hardware is needed to manage the power requirements of the TCBE residing on the motherboard. The required hardware design is likely to be complex, decreasing the cost efficiency.

In the second power model, the extra power source will decrease the cost efficiency more and the size of the PC module has to be increased to accommodate this extra hardware.

3. Conclusion

From the arguments we made it can be seen that there exists two possible solutions. One is the software solution of clearing the main memory by controlling the DRAM refresh rate by configuring the 443BX-chipset DRAM controller. The second one is the hardware solution of powering off the memory by making a simple direct hardware connection to the DIMM units.

The software-oriented solution is more flexible and it has an easier installation procedure than the hardware solution.

On the other hand there might be some problems with this scheme. One is bus arbitration. The configuration program may not start or complete if another program keeps or takes over the bus. The configuration sequence is a two-step process. First the configuration address register needs to be written and after that the data configuration register will be written. This sequence needs to be an atomic operation; otherwise malicious software might monitor this configuration attempt. When the malicious software sees that the address written into the configuration address register targets the DRAM memory controller refresh rate; it may intercept the configuration process. This way the configuration data may never be written with the zero refresh configuration value. The atomicity may be accomplished by checking on the PCI signals provided between the master and the target device. If a PCI transaction cannot complete, the reason that completion is prevented will be signaled to the master device. If a PCI configuration process is interrupted, the TCBE can check the specific status registers provided by the PCI architecture to assure that the transaction is completed. If there is any problem with the completion of the transaction, the TCBE will repeat the transaction until it gets completed successfully.

The hardware-based solution is not as flexible as the software-oriented procedure in the installation process. It needs extra installation effort and it may cause hardware problems if not installed appropriately. But it still has operational flexibility since the software can control it. This means that the hardware oriented refresh process can be initiated and ceased via software control.

The hardware solution is bulletproof. Since we bypass the whole PC system and directly control memory it cannot be circumvented by malicious code checking the PCI bus for a configuration attempt by the TCBE. Also this process is faster than the software version of it, as it has been noted earlier.

B. PENTIUM-II (400 MHZ) CACHE

1. An Introduction to Cache

Cache provides intermediate-level storage between system memory and the processor. It consists of a small amount of fast-access, but costly memory. In contrast DRAM system memory is both slower and larger. Cache is usually designed as static random access memory (SRAM).

Most programs contain code loops that are executed many times and have data structures that are accessed repetitively. In a situation like this the cache can dramatically improve the system performance by reducing the data or code access time for the processor. However if a program accesses code and data structures only once, there will be no improvement provided by the cache system.

2. Cache Operation

As described in [Ref. 8] and [Ref. 9], cache is made of SRAM, which is high cost, fast access memory. The cache controller keeps track of information, which has been copied into cache memory.

The cache architecture exploits two characteristics of most of the programs: temporal locality and spatial locality. Temporal locality addresses the notion that the longer it has been since information in the cache has been accessed, the less likely that information is to be used again. Spatial locality addresses the fact that programs are likely to need code or data that are close to locations already accessed.

When the processor initiates a memory read cycle the cache controller first checks the cache memory to determine whether the requested data exists in the cache or not. If a copy is present, it immediately reads the information from the cache and sends it back to the processor. This is called a *read hit*. This way the processor doesn't need to go to the system bus and this prevents wait states. The data transfer can be completed with zero wait states since the information is fetched from fast SRAM.

Usually cache works with the same speed as the core processor. If the cache controller determines that the data requested by the core processor doesn't exist in cache memory then the information must be read from DRAM memory, the system's main memory. This is known as a *read miss*. The read miss causes wait states since the access speed to DRAM memory is slower than the processor core speed.

When the requested information is sent from DRAM to the processor, it is also copied into cache memory by the cache controller. The speed up effect of the cache system becomes evident when programs, which have a lot of loops, are run. When a program executes many loops and makes many accesses to the same data structures, the system memory access time will dramatically reduce the speed of the program execution. The cache helps to avoid expensive DRAM accesses. The processor can access all the repetitive commands and data structures without needing any DRAM memory access.

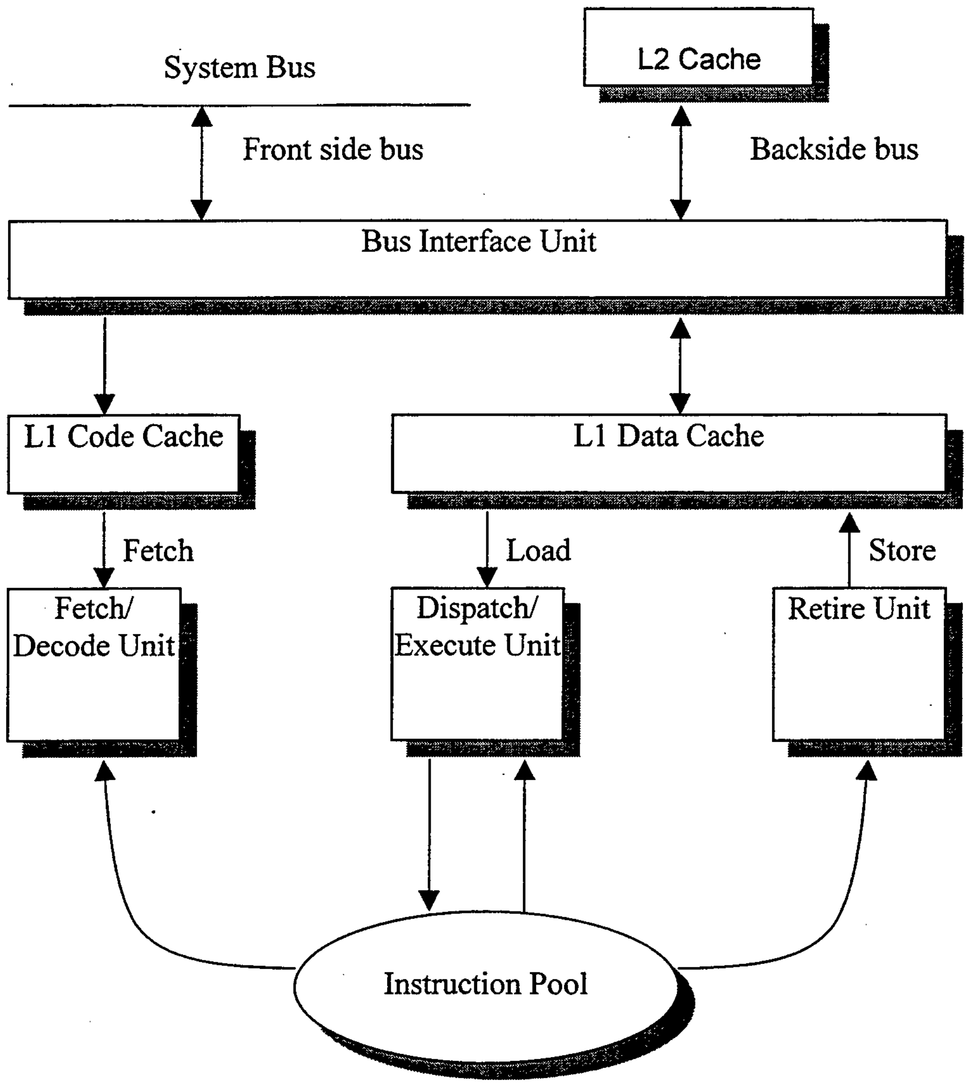

Figure 4.8 The Pentium II processor with cache and memory interfaces, after Ref. [17]

3. Cache Architecture in the Pentium-II System

The size and the characteristics of cache are machine-specific and may change from version to version. The Pentium II processor cache architecture consists of one internal 16KB Level-1 data cache, one internal 16KB Level-1 code cache and an external 512KB Level-2 unified cache connected to the processor by the backside bus.

The Pentium II processor can simultaneously transfer data on both the backside and front side (system bus) buses. The Intel Company calls this architecture as the Dual-Independent Bus Architecture (DIBA).

The backside bus provides a dedicated path between the core and the L2 cache. The Pentium II processor's backside bus is physically long. This slows down the backside bus speed. The backside bus operates at one half of the processor core speed. On the other hand, since the L1 cache capacity is doubled regarding to the Pentium Pro (P6) processor architecture, which reduces requirements for L2 level cache access.

The Pentium II processor also provides Error Correcting Code (ECC) protection both the L2 cache and the L1 caches.

The L1 data cache is 16KB, 4-way set associative with a 32-byte line size. This means that the cache look-up process is done as a set of four cache entries each of which is 32-byte in size. The L1 code cache is 16KB, 4-way set associative with a 32-byte line size. The unified, L2 cache is 512KB, 4-way set associative with a 32-byte line size. The cache structure uses a look-through type of caching. This means that the processor will first check the cache structure and then it will decide to access the system memory since the address that it has been looking for doesn't exist in the cache memory. The simplified block diagram of the Pentium-II processor cartridge is as given below.

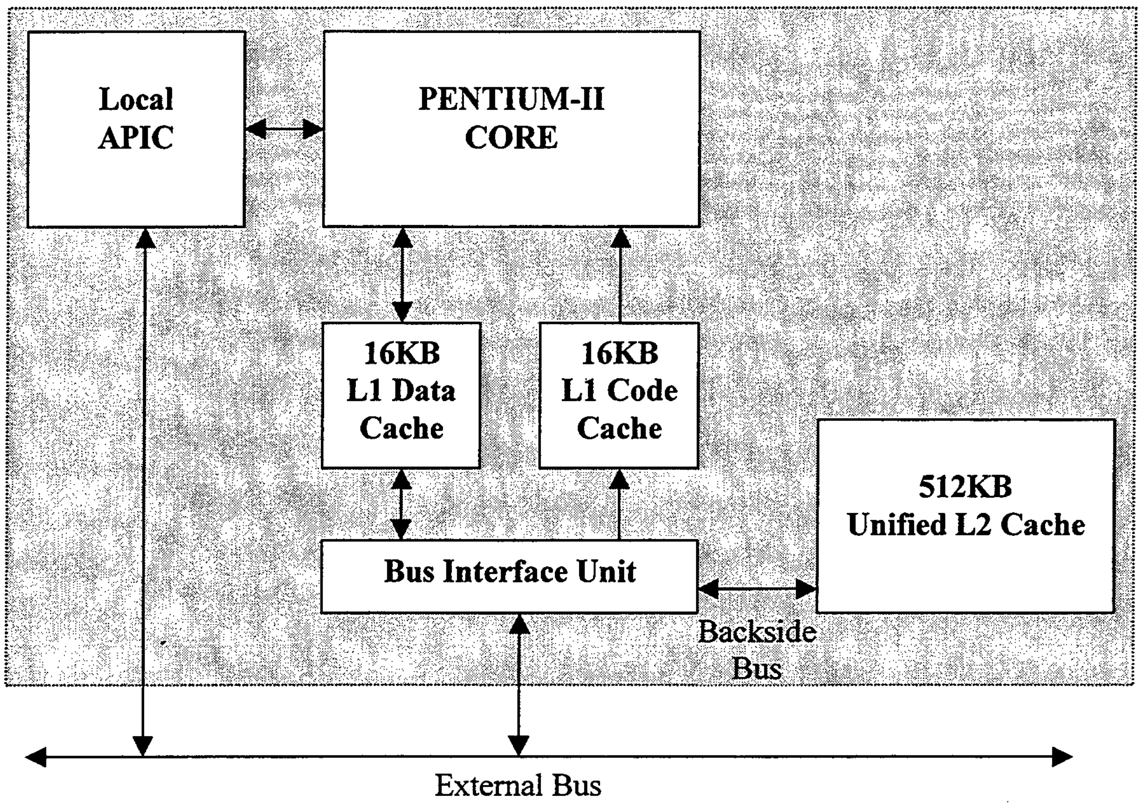

Figure 4.9 Simplified Logical Block Diagram of the Pentium-II Processor Cartridge, from Ref. [8]

The algorithm used for the replacement of the lines in the cache is a pseudo least recently used (LRU) algorithm. Simply put, the lines which are used the least are replaced when there is a need for a new space in the cache structure.

The L2 cache is physically consisted of five SRAM modules. These modules are on the same card, called the substrate, with the processor. There are four SRAM data modules and one L2 cache tag SRAM module.

4. Need for Object Reuse Control on Cache

In an MLS system resources are shared under the control of the system. Object reuse must be controlled on the client system. The cache memory is one of the main storage objects in the client computer. We want object reuse control over the cache to prevent any sensitive information leakage between two sessions.

5. Object Reuse Control of Cache

In this section possible ways of removing the information stored in the cache between sessions are discussed. The concepts are derived from the information documented in the Pentium II publications from the Intel Company. Other object reuse techniques for cache might be possible but detailed knowledge of the cache from Intel-internal documents would be needed and this information is proprietary.

a. Cache Overwrite

(1) Concept of technique: After the main system memory is cleared by the TCBE, the CPU needs to be restarted since it will have been halted. When the reset signal is applied to the Pentium-II by the TCBE to restart the halted CPU all cache data is marked as invalid.

The TCBE can make the CPU run a program, which accesses memory locations non-repetitively, which means that each memory read access must be done to each memory address location only once. Since the cache system is based on the temporal locality and spatial locality principles the cache controller immediately fetches information accessed by the processor into cache memory. At that moment the memory contains meaningless random values because it has been cleared by the TCBE. This way the old cash can be overwritten by random values from the memory, leaving the cache memory useless for the information leaking attempts.

In the same way the memory can be written with a known bit pattern such as all ones or all zeros. This way the cache can be filled with predefined bit pattern.

(2) Advantages and disadvantages: The advantage of this system is that the cache memory can be initialized with known values such as all ones or zeros so that the system always starts with a known cache state. Making the Pentium II processor write a known data pattern into the memory can accomplish this process.

On the other hand, a disadvantage of this method is that overwriting the memory is time consuming and it will cause delays between session changes. The cache overwriting process will be slow since the Pentium-II cache utilizes look-through type of caching. In a look through type design the host processors memory access requests are first submitted to the look through cache to determine if copy of the target address line exists in the cache. This will incur a look up penalty, in the event of a cache miss, causing an extra delay.

b. Cache Clearing

(1) Concept of technique: The caches can be cleared by using the Invalidate Internal Caches (INVD) command. This command flushes the processor's internal caches, the L1 code cache and the L1 data cache. It also issues a special-function

bus cycle, which also flushes the external L2 cache. The data held in the internal caches will not be written back to main memory. The INVD instruction is a privileged instruction. The current privilege level of the program or procedure must be 0 to execute this instruction. Data cached internally and not written back to memory will be lost.

(2) Advantages and disadvantages: The advantage of this method is that it won't take as much time as it takes to write into memory. This process is initiated by a command, INVD, and the rest of the process is completed by the hardware.

One side effect of this method is that it is not clear what the state of data in the cache will be after the cache is flushed by the INVD command. It could be zeros or all ones or both of them in random order in cache memory. The Intel literature doesn't document the state of the cache memory after the INVD command is executed but does state that all the data in the cache memory will be lost after the execution of the INVD command.

On the other hand there might be some side effects with this scheme. Intel states that the INVD command will cause the data in the cache to be cleared. However it is not clearly documented what the data values in the cache will be after this flushing process. If the state of the cache needs to be known, the proposed methods in section one and section two can be executed together. First the INVD command can be used to flush the cache and then the memory writing process can be accomplished to overwrite the cache, so that the cache data values can be set to a known state of value.

We need to be sure that either the cache is really overwritten or the INVD command is really executed by the CPU. These processes should be initiated by the TCBE, so it has to be the bus master of the PCI transaction for these processes. PCI bus ownership must be guaranteed to ensure a continuous transaction. If the transaction is interrupted by malicious software the TCBE will know that the PCI transaction couldn't be completed and it must retry the transaction. A single atomic total transaction for object reuse control would be appropriate.

6. Conclusion

From the arguments above, it can be seen that there exist two possible solutions to provide object reuse control over cache. One is to overwrite the cache by making the Pentium II processor write to system main memory. The second is to directly flush the cache memory by making the Pentium II processor execute the INVD command.

The second solution is faster than the first one since there is no system main memory access required to overwrite the cache. This process can be accomplished over the backside bus and the internal bus of the Pentium II processor but not the system PCI bus, which is relatively slower.

C. OBJECT REUSE CONTROL OF ACCELERATED GRAPHICS PORT AND REAL TIME CLOCK RAM

Another two of the significant storage areas in the PC are the accelerated graphics port (AGP) RAM and the real time clock (RTC) RAM. In this chapter each of these storage areas will be investigated to find a solution for their object reuse.

1. AGP RAM

The AGP interface for the PC passes through the 82443BX Host/PCI Bridge. The 82443BX doesn't support the existence of any other I/O devices beside itself on the CPU bus. This means that all I/O requests passing through the 82443BX are controlled by this chipset.

The 82443BX generates either PCI or AGP bus cycles for all I/O accesses initiated by the CPU. The I/O accesses, other than ones used for PCI configuration space access, are normally forwarded to the PCI bus unless they fall within the PCI-1/AGP I/O address range. When the CPU initiates an I/O cycle targeting the AGP I/O address range, the 82443BX directs these non-memory (I/O) accesses to the AGP bus interface.

The PCI interface for an AGP card residing on the PCI bus is as follows. The 82443BX accepts all memory-read and write accesses to main DRAM. The memory-write accesses to the AGP memory range are acknowledged, however, the 82443BX will not respond to memory read accesses in this range. Memory-read and write accesses are allowed to the Graphics Aperture, which is located in the system main memory. PCI accesses that fall elsewhere within the PCI memory range will not be accepted.

There are separate rules for AGP interface decoding regarding the AGP card residing on the AGP bus. If the cycles are initiated by using the PCI protocol on the AGP bus, accesses between AGP and PCI devices are limited to memory writes. Write cycles are forwarded to the PCI bus if the addresses are not within main DRAM range, AGP memory range or Graphics Aperture range. The 82443BX chipset claims AGP initiated memory read transactions decoded to the main DRAM range or the Graphics Aperture range, both of which reside in the system main memory. All other memory read requests would be master-aborted by the AGP initiator as a consequence of the 82443BX not responding to a transaction. If an agent on AGP issues an I/O, PCI Configuration or PCI Special Cycle transaction, the 82443BX chipset will not respond and the cycle will result in a master-abort.

For cycles initiated by using the AGP protocol, all must reference main memory range, main DRAM address range or Graphics Aperture range. Graphics Aperture range is also physically mapped within DRAM, but it uses a different address range. In this transaction, the bus master is the AGP accelerator.

Consider the scenario for malicious code trying to pass information from one session to the other by using the AGP RAM. The code will try to write the sensitive data into AGP RAM. In a new session, it will try to extract the data from the AGP RAM. Consider the possible actions that the malicious code can take. It can write into the AGP RAM by via the PCI protocol. In the AGP protocol, the only master is the AGP card, so the malicious code would have no control of any read or write process, assuming that the AGP card is a good and trusted card. When the malicious code tries to read from the AGP RAM via PCI protocol the transaction will be master aborted since only main memory range or Graphics Aperture range, also physically mapped within main memory, read access is allowed by the 82443BX chipset.

When the 82443BX Host/PCI AGP bridge access decode rules are analyzed for an AGP card residing on the AGP bus, it can be concluded that a PCI bus master can not accomplish a successful read access to the target AGP card, by using either PCI or AGP access protocol. Thus, it can be said, that even if a malicious program writes sensitive data into the AGP local memory region, this information cannot be read back out, thereby eliminating object reuse considerations.

Malicious code may write data into the AGP local memory in an appropriate format with the intent for putting this sensitive information on the client display unit. Good protection against this threat would be to flush the displayed sensitive information by writing predefined display data to the whole local memory buffer from the TCBE at the beginning of the new session. This way, the old display values would never be seen in the new session.

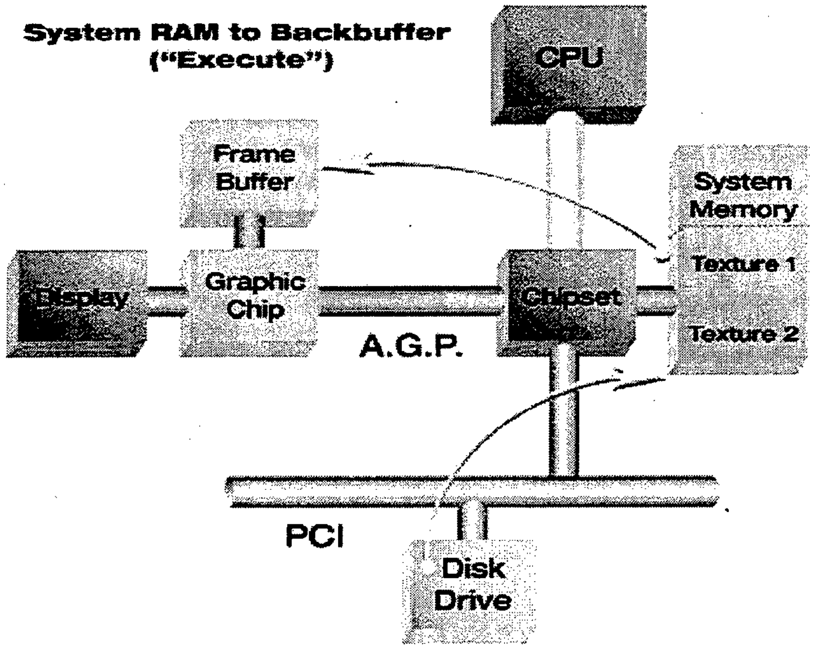

Figure 4.10 The generic AGP bus relation with the rest of the system from Ref. [53]

2. RTC RAM

The Real Time Clock (RTC) module is located in the 82371AB (PIIX4) PCI/ISA accelerator and provides a date-and-time keeping device with alarm features and battery backed-up operation. The position of the PIIX4 in the logical organization of the PC system is shown in Figure 4.11.

The RTC counts seconds, minutes, hours, days, and day of the week, date, month and year with leap year compensation.

The RTC module contains 256 bytes of battery-backed static RAM (SRAM) in two banks, namely, the standard bank and the extended bank. The first 10 bytes of the standard bank contain the time and date information. The next 4 bytes are used as four control registers (A, B, C, and D) to control the operation of the RTC. The rest of the 114 bytes are used as general purpose RAM. The extended bank has 128 bytes used as general purpose RAM. These general-purpose storage areas are used for preserving the configuration information such as boot order information, device configuration data, etc.

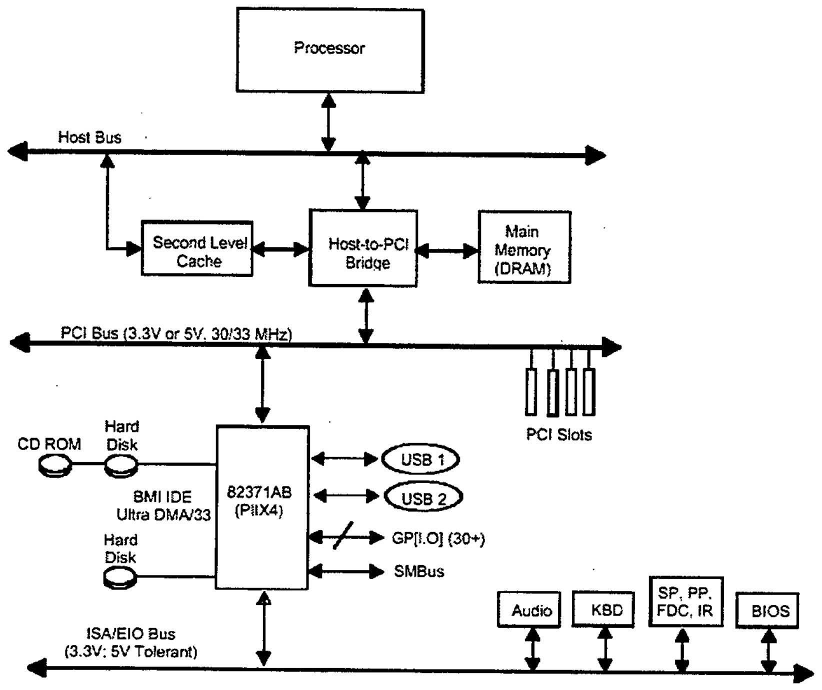

Figure 4.11 Position of the PIIX4 in the logical organization of the PC from Ref. [5]

Time, calendar and alarm can be represented in either binary or Binary Coded Decimal (BCD) format. The hour can be represented in 12 or 24-hour format. The RTC module requires an external oscillating source of 32.768 KHz. This clock signal is divided down to 1 Hz. Signal.

All data movements between the host CPU and RTC are done through registers mapped to the ISA I/O space at locations 70-73h. The standard RAM bank is accessed through the registers at I/O addresses of 70h and 71h. For the extended RAM bank, the ISA I/O address 72h is used as the address pointer and ISA I/O address 73h is used as the data register. Only PCI masters can access the internal registers. ISA master access is not supported.

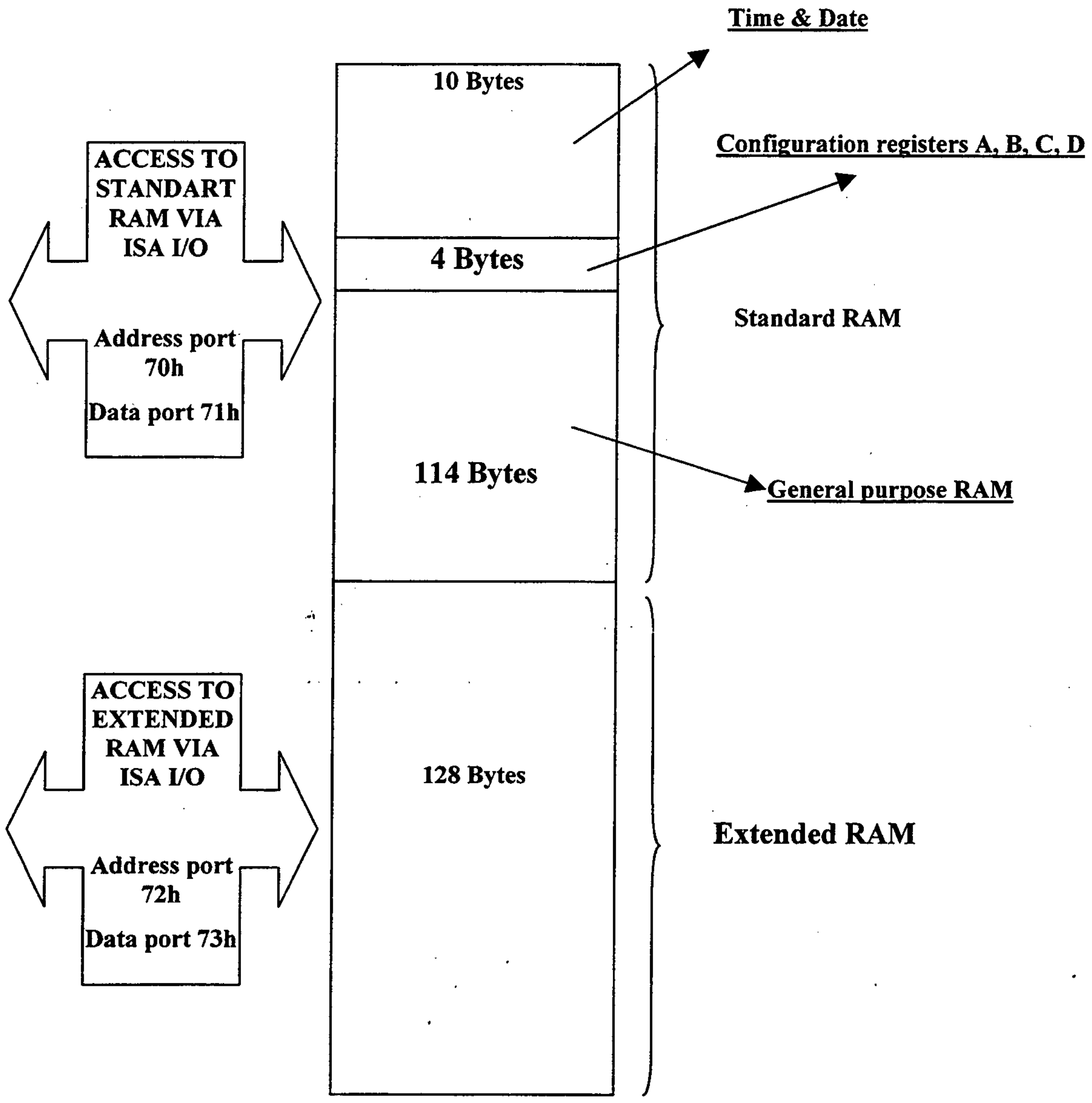

Figure 4.12 The access ports for the RTC module

An update cycle occurs once every second. During this procedure the stored time and date are incremented, overflow checked for the upper limit number value, a matching alarm condition is checked, and the time and date are rewritten to the RAM locations.

The real time clock configuration register (RTCCFG) is used to configure the internal real time clock. The first and the third bits of this register are used to configure the availability of the RTC RAM.

The RTC battery backed RAM supports two 8-byte ranges that can be disabled via RTCCFG. In this way, these memory locations cannot be readable or writable. The same enable and disable process can be done to the standard and extended memory ranges. A write cycle to these locations has no effect. A read cycle to these locations does not return the actual location value. RTCCFG is a write-once register. Once enabled anytime after the boot process, this function can only be disabled by a hard reset. It is not possible to reset this register by software means.

As the first solution for providing object reuse control for this memory region, the TCBE can read RTC and configuration values into its memory and then lock the standard and extended RAM banks before the operating system is given control. This way any write or read attempt to the RTC RAM can be prevented. After this, the TCBE may provide the system with the time and configuration values. This can be done by running a daemon watching the ISA I/O address references and then emulating the ports. This requires the TCBE to watch over the O/S at all times. If the TCBE has a snooping capability as the CPU does, then this may be done. This way, any read access targeting the specific port addresses can trigger the TCBE and the TCBE may provide the requested values. This may not be possible with the current commercial PCI add-on cards but this idea will provide another insight for the solution of this problem and it may be realized in the future.

The second solution for object reuse control is that RTC memory can be read back to the TCBE and after the session the read information can be written back to RTC. This way any possible sensitive information kept in the RTC RAM would be overwritten. After the write back process, the RTC RAM should be locked so that a malicious program can do no re-writing.

Even if the configuration information were changed during the session, overwriting these configuration data would not be a problem since every new session will start with a pre-defined configuration. On the other hand, the write back process should be atomic so that it can be completed uninterrupted. This way we can be sure that we have really accessed and overwritten the whole RTC RAM.

As the third solution, with the time information excluded, RTC memory can be checksummed at the beginning of each session and the checksum value can be kept in the TCBE. At the end of a session, comparing the checksum value can ensure the integrity of the RTC RAM area. If the checksum value doesn't compare, then the TCBE can give a warning and freeze the system or can overwrite the RTC RAM with the appropriate configuration values. This action can be determined according to the security policy enforced by the system.

The fourth solution differs slightly from the third, and requires that the default or approved RTC RAM values or checksum be kept in the TCBE at all times. This way the RTC RAM area can be checked at the end of the session. This method prevents extra checksum calculations at the beginning of every new session. Since there is not checksum process involved at the beginning of the session, the new session establishment can be faster.

To conclude, using the second and fourth solutions would yield the desired object reuse control.

V. EXPERIMENTATION

In this chapter we will explain an experiment to prove the feasibility of solution number two, TCBE read and write access to RTC RAM, and solution number four, RTC RAM checksum control by TCBE, which are discussed in chapter six section-B.

A. HYPOTHESIS

The Intel i960 PCI board, the prototype TCBE, can access the RTC RAM and commence read and write operations on this memory region.

B. DESIGN OF EXPERIMENTATION

The Intel i960 is a PCI board, which can access the PCI bus of the client PC. This board can initiate PCI transactions as a bus master on the PCI bus of the client. The RTC RAM, on the other hand, resides on the ISA bus hosted by the south bridge (PIIX4). The logic diagram for the relation between the PCI and ISA buses is given in Figure 4.11.

The purpose of this experiment is to reach the PCI bus as the bus master via the i960 board, then to go through the ISA bus via the south bridge, and access the RTC RAM.

C. IMPLEMENTATION

First we need to initialize the input/output configuration values. Then we will access one of the locations in RTC RAM. We will read the byte value of that location and store it in one of the global registers in the i960 board, thereby showing that the TCBE will be able to read and store the configuration values from the RTC RAM. Following this simulation, we will write to that address location in RTC RAM as if malicious code in the PC had written to the RTC RAM in order to leak sensitive information to another session level. Then, we will read back the value of the RTC RAM, simulating that the TCBE reads the data on the RTC RAM in order to perform integrity check such as comparing, check-summing etc. Next, we will write the original value, which was kept in i960, back to the memory location in RTC RAM, see Figure 5.1. This simulates that the TCBE overwrites the sensitive information rendering the efforts of the malicious code worthless.

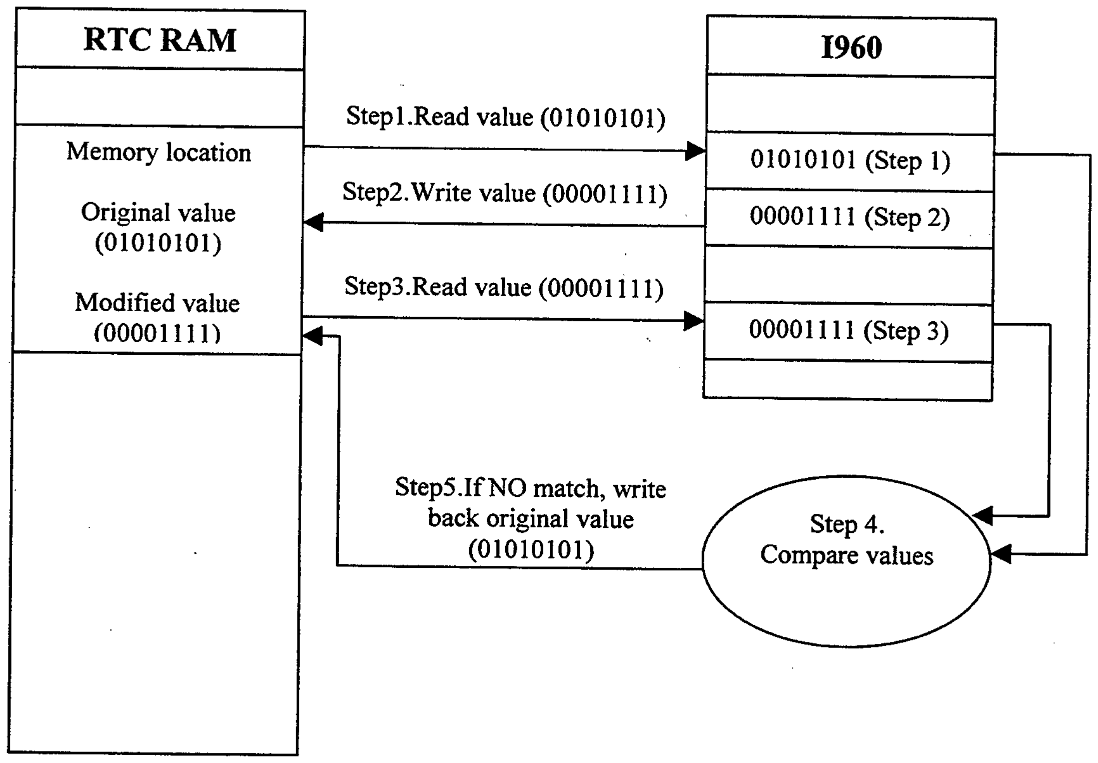
RTC RAM
Memory location
Original value (01010101)
Modified value (00001111)
I960
Step1.Read value (01010101)
01010101 (Step 1)
Step2.Write value (00001111)
00001111 (Step 2)
Step3.Read value (00001111)
00001111 (Step 3)
Step 4. Compare values
Step5.If NO match, write back original value (01010101)

Figure 5.1 The flow of the experiment

The assembler code, which accomplishes these actions, is given in Table 5.1

```
.text

.globl _main

_main:

lda 0x90000070, g0    /* loads the address port value for I/O */

lda 0x90000071, g1    /* loads the data port value for I/O */

lda 0x0000000e, g2    /* loads the byte location of the RTC RAM value */

stob g2, (g0)         /* stores the location of the RTC RAM value into the I/O
                         address port */

ldob (g1), g3         /*gets the original value in the RTC RAM from the data
                         port */

 lda 0x00000099, g5   /* loads the value to overwrite the original data into the
                         RTC RAM from the data port */

stob g2, (g0)         /* stores the location of the RTC RAM value into the I/O
                         address port */

stob g5, (g1)      /*change the original RTC RAM value to 99h*/

stob g2, (g0)     /* stores the location of the RTC RAM value into the I/O address
                     port */

ldob (g1), g6    /* gets the modified value in the RTC RAM from the data port */
```

```
stob g2, (g0)       /* stores the location of the RTC RAM value into the I/O
                    address port */

stob g3, (g1)      /*writes the original value back to RTC RAM */

stob g2, (g0)    /* stores the location of the RTC RAM value into the I/O address
                 port */

ldob (g1), g6      /*gets the original value from the data port for confirmation*/

forever:

b forever        /*loops forever */
```

Table 5.1 Assembler code for simulation of object reuse control in RTC RAM

D. EXPERIMENTATION DATA

For this experiment, the required data values are as given below:

The Primary outbound I/O window address for the RTC RAM address port: 0x90000070

The Primary outbound I/O window address for the RTC RAM data port: 0x90000071

The RTC RAM memory location address: 0x0000000E

E. CONCLUSION

It is possible to access RTC RAM memory from the TCBE by implementing the code described above. The access to RTC RAM will provide the TCBE with the ability to control the contents of this memory region. By controlling this memory region, the TCBE will also have object reuse control of this battery backed non-volatile storage area in the client PC.

In addition, there is an Address Translation Unit Status register embedded in the i960 board. This status register can be used to detect any interruptions of the I/O cycle to provide atomicity to the transaction.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

A. SUMMARY

The main purpose of this study is to contribute to the realization of a multilevel secure local area network (MLS-LAN). The system consists of a high assurance, Trusted Computing Base (TCB) that acts as a server. Clients consist of COTS workstations and software, augmented with a Trusted Computing Base Extension (TCBE). The Object reuse mechanisms are designed to assure that the user (subject) of the system doesn't obtain residual information from system resources.

In chapter one, we introduced object reuse and the need for object reuse in secure systems. In chapter two, we investigated the storage areas on the SOYO SY-6BE+ type motherboard, which hosts an Intel Pentium II 400MHz CPU and Intel 440BX AGPset consisting of the 82443BX Host Bridge and the 82371EB PIIX4E. With this investigation we determined the devices with storage areas and we picked the main storage areas for object reuse control analysis.

In the third chapter, we walked through the PC boot process and reviewed important issues in the PCI protocol. These will be helpful in finding solutions to provide object reuse control for the main storage areas.

In the Chapter IV, we proposed and evaluated possible solutions for the object reuse control of main storage areas in the PC.

In the fifth chapter, we provide detailed information about an experiment that was conducted in order to prove the feasibility of one of the proposed solutions for object reuse control of RTC RAM.

B. RECOMMENDATIONS FOR FUTURE WORK

In the future, efforts to accomplish the miniaturization, for example VLSI level design, of selected TCBE components or functions may be considered.

In this study we did not investigate areas with small amounts of storage areas, for example those just a couple bytes in size. These include the internal CPU registers and the registers on other chips such as W83781D / 836AC Winbond Hardware Monitoring IC.

We couldn't investigate certain elements because of proprietary or unavailable documentation. In this study, we pointed out these elements for future object reuse control efforts.

The behavior of the PC needs to be experimented with when the memory is cleared or overwritten. Will the CPU halt, and if it halts, can it be recovered by a reset signal applied by the TCBE? If the CPU halts, will the DRAM controller be functional and allow the TCBE access to main memory to ensure that main memory has been really erased? While the CPU is halted, can the TCBE access the CMOS RAM and accomplish read and write operations? Are there any side effects caused by clearing the memory DIMM modules by making a direct hardware connection from the TCBE, as it is proposed in Section 2.a in Chapter IV? Can the i960 board be connected to an external power source other than the one provided via PCI interface? These questions can also be addressed in future studies.

Our study can provide a basis for future object reuse control studies. It can provide insights for new hardware or software designs that have built-in object reuse control, such as new trusted operating systems or object reuse free motherboards and compatible peripherals such as sound cards, AGP cards, SCSI cards etc.

C. CONCLUSION

In this study we proposed different object reuse control solutions for different devices and we proved the feasibility of one proposed solution. This study establishes a foundation for object reuse control efforts targeting COTS PC products manufactured by various vendors. This study also provided information for the design specifications of the TCBE and will hopefully lead to the use of highly secure systems with low cost and easy installation throughout the government and military services.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. DESIGN CALCULATIONS FOR ZERO REFRESH

The derivation of the required signals to design the logic circuit for providing the zero-refresh rate is given below

NAME (FUNCTION)	CS#	RAS#	CAS#	WE#	DQM	ADDR	DQs
COMMAND INHIBIT (NOP)	H	X	X	X	X	X	X
NO OPERATION (NOP)	L	H	H	H	X	X	X
ACTIVE (Select bank and activate row)	L	L	H	H	X	Bank/Row	X
READ (Select bank and column, and start READ burst)	L	H	L	H	L/H[8]	Bank/Col	X
WRITE (Select bank and column, and start WRITE burst)	L	H	L	L	L/H[8]	Bank/Col	Valid
BURST TERMINATE	L	H	H	L	X	X	Active
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	X	Code	X
AUTO REFRESH or SELF REFRESH (Enter self refresh mode)	L	L	L	H	X	X	X
LOAD MODE REGISTER	L	L	L	L	X	Op-Code	X
Write Enable/Output Enable	–	–	–	–	L	–	Active
Write Inhibit/Output High-Z	–	–	–	–	H	–	High-Z

Table A.1 The truth table for SDRAM commands from Ref. [23]

Note: H = logic 1, L = logic 0, X = logic 1 or logic 0

When the DRAM is in power down state, no refreshing process takes place. If DRAM remains in this state for more than 64 milliseconds, data loss occurs.

The power down state results from either of two conditions. First, when the DRAM control logic unit (see Figure 4.4) receives clock enable signal low (CKE = 0) along with the No Operation (NOP) command. Second, when the DRAM control logic unit receives CKE signal low (logic 0) along with the Command Inhibit (CI) command.

Table A.1 shows the signal combinations required for initiating the CI and NOP commands. So we need to make a logic analysis to determine the simplified logical function in order to put the DRAM into power down state.

The logical expression of the requirements for the power down state is as follows:

F = Power down

N = NOP command

C = CI command

A = CKE, B = CS#, C = RAS#, D = CAS#,

E = WE#, F = DQM#, G = ADDR, H = DQs

F = N + C

$$N = \underline{A}\,(\underline{B}\,(C+\underline{C})\,(D+\underline{D})\,(E+\underline{E})\,(F+\underline{F})\,\overbrace{(G+\underline{G})}^{\text{Logic 1}}\,(H+\underline{H}))$$

$$N = \underline{A}\,\underline{B}$$

$$C = \underline{A}\,(B\,(C+\underline{C})\,(D+\underline{D})\,(E+\underline{E})\,(F+\underline{F})\,(G+\underline{G})\,(H+\underline{H}))$$

$$C = \underline{A}\,B$$

$$F = N + C = \underline{A}\,\underline{B} + \underline{A}\,B = \underline{A}\,(\underline{B}+B) = \underline{A} = \underline{CKE}$$

After we simplified the logic function we determined that to put the DRAM into a power down state, the TCBE is required to provide a logic zero signal to the CKE pin (pin 37) [Ref. 23] of the DRAM module.

LIST OF REFERENCES

1. Croucher, Phil, *The BIOS Companion*, Advice Press, Palo Alto, CA, December 1998.

2. Barry, Kauler , *Windows Assembly Language and Systems Programming*, R&D Books, Emeryville, CA, 1997.

3. Intel, *Intel 440BX AGP set design guide*, Intel corporation, Mt. Prospect, IL, April 1998.

4. Intel, "82443 BX Host/PCI AGP Controller", www.intel.com, April 1998.

5. Intel, "82371 AB", www.intel.com, April 1997.

6. Shanley, Tom, *Plug and play system architecture*, MINDSHARE, INC., Reading, Massachusetts, August 1999.

7. Shanley, Tom, *protected mode software architecture*, MINDSHARE, INC., Reading, Massachusetts, December 1998.

8. Shanley, Tom, *Pentium pro and Pentium II system architecture*, MINDSHARE, INC., Reading, Massachusetts, September 1999.

9. Shanley, Tom, and Anderson, Don, *Pentium processor system architecture*, MINDSHARE, INC., Reading, Massachusetts, January 1998.

10. Shanley, Tom, and Anderson, Don, *PCI System Architecture*, MINDSHARE, INC., Reading, Massachusetts, September 1999.

11. Solari, Edward, and Willse, George, *PCI Hardware and Software Architecture & Design*, Annabooks, San Diego, CA, November 1998.

12. Shanley, Tom, and Anderson, Don, *ISA system architecture*, MINDSHARE, INC., Reading, Massachusetts, September 1999.

13. Dzatko, Dave, *AGP system architecture*, MINDSHARE, INC., Reading, Massachusetts, December 1998.

14. Intel, *Intel architecture software developer's manual volume-1*, #243190-001, Intel Corporation, Mt. Prospect, IL, 1997.

15. Intel, *Intel architecture software developer's manual volume-2*, #243191-001, Intel Corporation, Mt. Prospect, IL, 1997.

16. Intel, *Intel architecture software developer's manual volume-3*, #243192-001, Intel Corporation, Mt. Prospect, IL, 1997.

17. Intel, *Pentium II Processor Developer's Manual*, #243502-001, Intel Corporation, Mt. Prospect, IL, 1997.

18. Intel, *Pentium II Processor Specification Update*, #243337-029, Intel Corporation, Mt. Prospect, IL, August 1999.

19. Holtek Semiconductor Inc., "HT27C512 OTP CMOS 64K′ 8-Bit EPROM", www.holtek.com.tw, May 1999.

20. Holtek Semiconductor Inc., "HT82V167/HT82V168 VCD$^+$ A/V DECODER", www.holtek.com.tw, March 1999.

21. Intersil, "HA-5002", www.intersil.com, November 1998.

22. National Semiconductor Company, "LM2635", www.national.com, November 1999.

23. Micron Semiconductor Products Inc., "MT48LC8M8A2", www.micron.com/mti/msp/html/datasheet.html, November 1999.

24. Fairchild Semiconductor, "74F174", www.fairchildsemi.com, July 1999.

25. Philips Semiconductors, "74HC/HCT74", www-us.semiconductors.philips.com, February 1998.

26. Philips Semiconductors, "74LV244", www-us2.semiconductors.philips.com, May 1998.

27. Fairchild Semiconductor, "DM7407", www.fairchildsemi.com, March 1998.

28. Fairchild Semiconductor, "DM74ALS00A", www.fairchildsemi.com, February 1998.

29. Fairchild Semiconductor, "DM74ALS05A", www.fairchildsemi.com, February 1998.

30. Fairchild Semiconductor, "DM74ALS08", www.fairchildsemi.com, February 1998.

31. National Semiconductor Company, "LM555/LM555C", www.national.com, May 1997.

32. Motorola, "MC54/74HCT14A", www.motorola.com, October 1995.

33. Cypress Semiconductor Corporation, "W40S11-23", www.cypress.com, September 1999.

34. Winbond Electronics Corporation, "W83781D", www.winbond.com.tw, November 1997.

35. Integrated Technology Express,Inc., "IT8687R", www.ite.com.tw, March 98.

36. Integrated Technology Express,Inc., "IT8671F/ IT8671RF/ IT8671R", www.ite.com.tw, March 98.

37. Intel, "Pentium II Processor at 350 MHz, 400MHz, and 450MHz", www.intel.com, 1998.

38. Phoenix Technologies Ltd., "CMOS Setup Utility User's Guide for Intel 82440BX AGP set", www.phoenix.com, January 1999.

39. Phoenix Technologies Ltd., "Award Bios Version 4.51PG Post Codes & Error Messages", www.phoenix.com, January 1999.

40. Phoenix Technologies Ltd., "Phoenix Bios 4.0 Release 6.0.51PG Post Tasks & Beep Codes", www.phoenix.com, 1997.

41. Catalyst Semiconductor Inc., "CAT93C46/56/57/66/86", www.catsemi.com, February 1998.

42. Linfinity, "LX5115", www.linfinity.com, November 1999.

43. Intel, "ATX Specification", www.intel.com, 1998.

44. Soyo Computer Inc, "6BE+ 82440 BX PCI Mainboard Users Guide & Technical reference", www.soyo.com.tw, April 1998.

45. Soyo Computer Inc, "6BE+ 82440 BX PCI Quick Start Guide", www.soyo.com.tw, September 1998.

46. Intel, "Accelerated Graphics Port Interface Specification", www.intel.com, May 1998.

47. Intel, "I960 RM/RN I/O Processor Developer's Manual", www.intel.com, July 1998.

48. Intel, "I960 Processor Assembler User's Guide", www.intel.com, December 1997.

49. Intel, "80960RM I/O Processor", www.intel.com, August 1998.

50. Intel, “gdb960 User’s Manual”, www.intel.com, December 1997.

51. *Common Criteria for Information Technology Security Evaluation (CC 2.1)* CCIMB-99-031, August 1999.

52. *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, National Computer, Security Center, December 1985.

53. Intel, “AGP Technology Tutorial”, www.intel.com, February 2000.

54. Cadence, “Cadence”, www.cadence.com, Cadence Design Systems, February 2000.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., Ste 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943-5193	1
4.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
5.	Dr. Cynthia E. Irvine Computer Science Department Code CS/Ic Naval Postgraduate School Monterey, CA 93943-5193	3
6.	Dr. William A. Arbaugh WAA Associates, LLC. 4264 Hermitage Dr. Ellicott City, MD. 21042	1
7.	Mr. James P. Anderson James P. Anderson Company Box 42 Fort Washington, PA 19034	1
8.	Mr. Paul Pitelli National Security Agency Research and Development Building R2, Technical Director 9800 Savage Road Fort Meade, MD 20755-6000	1

9. Dr. Lee Taylor 1
National Security Agency
Research and Development Building
R22, Chief
9800 Savage Road
Fort Meade, MD 20755-6000

10. Ms. Donna Belt 1
National Security Agency
Research and Development Building
R23, Chief
9800 Savage Road
Fort Meade, MD 20755-6000

11. CAPT Dan Galik 1
Space and Naval Warfare Systems Command
PMW 161
Building OT-1, Room 1024
4301 Pacific Highway
San Diego, CA 92110-3127

12. Commander, Naval Security Group Command 1
Naval Security Group Headquarters
9800 Savage Road
Suite 6585
Fort Meade, MD 20755-6585

13. Ms. Deborah M. Cooper 1
Deborah M. Cooper Company
P. O. Box 17753
Arlington, VA 22216

14. Ms. Louise Davidson 1
N643
Presidential Tower 1
2511 South Jefferson Davis Highway
Arlington VA 22202

15. Mr. William Dawson 1
Community CIO Office
Washington DC 20505

16. COL Timothy A. Fong 1
COMMANDER/JED/IAESO
Defense Information Systems Agency
Columbia Pike Offices
5600 Columbia Pike
Falls Church, VA 22041-2717

17. Mr. David R Basel 1
Chief, IA Infrastructure and Engineering Branch
Information Assurance Support Organization
DISA/D6/JED
5600 Columbia Pike
Falls Church, VA 22041

18. LCDR James P. Downey 1
DISA D6/IAESO/MSL Engineering
5600 Columbia Pike
Falls Church, VA 22041-2717

19. Deniz Kuvvetleri Komutanligi 1
Personel Tedarik ve Yetistirme Daire Baskanligi
06100 Bakanliklar, Ankara
TURKEY

20. Cihan Agacayak 3
PK. 71
06502 Bahcelievler, Ankara
TURKEY

21. Deniz Harp Okulu Komutanligi 2
Kutuphane
81704 Tuzla, Istanbul
TURKEY